December 1978

NASA CR 158970-4

SPAR STRUCTURAL ANALYSIS SYSTEM

REFERENCE MANUAL

SYSTEM LEVEL 13A

Volume 4

PROGRAMMER REFERENCE

By

C.E.Jones and W.D.Whetstone

Prepared under Contract NAS1-14464

By

ENGINEERING INFORMATION SYSTEMS, INC.
5120 CAMPBELL AVENUE, SUITE 240
SAN JOSE, CALIFORNIA 95130

For

NASA
National Aeronautics and
Space Administration

FOREWORD

This document was prepared under contract NAS1-13977, through subcontract LL90A1760K with Lockheed Missiles & Space Company, Inc.; funded by the Langley Research Center of the National Aeronautics and Space Administration. The Contracting Officer's Technical Representative was J. C. Robinson.

The purpose of this document is to describe the internal logic of designated areas of SPAR system code. In anticipation of future additions, the following organization has been chosen: the document will consist of a sequence of named chapters, with each chapter incorporating its own table of contents and page/section notation. Chapters are tabulated in a General Table of Contents.

Submitted by
Engineering Information Systems, Inc.

W. D. Whetstone
President

GENERAL TABLE OF CONTENTS

The K2D Series

TABLE OF CONTENTS

GENERAL INFORMATION

Readers of this section should have a thorough understanding of the assumed stress field-minimum complementary energy method for computing finite element stiffness and stress matrices. This method was originally proposed in 1964 by T.H.H. Pian, Reference 1*. The basic theoretical procedure and the assumptions associated with the SPAR/K2D implementation of the Pian method are presented in Reference 2*. This section documents the K2D-series of subroutines for computing stiffness and stress matrices for two-dimensional finite elements.

K2D can accommodate the following:

- 3- and 4-node element geometries,
- membrane, plate bending, uncoupled membrane + bending, and coupled membrane + bending constitutive relations, and
- quadrilateral elements with node 4 positioned slightly outside the x-y plane of the element reference frame.

As discussed in References 1 and 2, the following three principal computational tasks are required:

1. Compute the internal strain energy matrix, H, such that

$$V = \tfrac{1}{2} B^* H B,$$

2. Compute the boundary work matrix, T, such that

$$W = B^* T q, \text{ and}$$

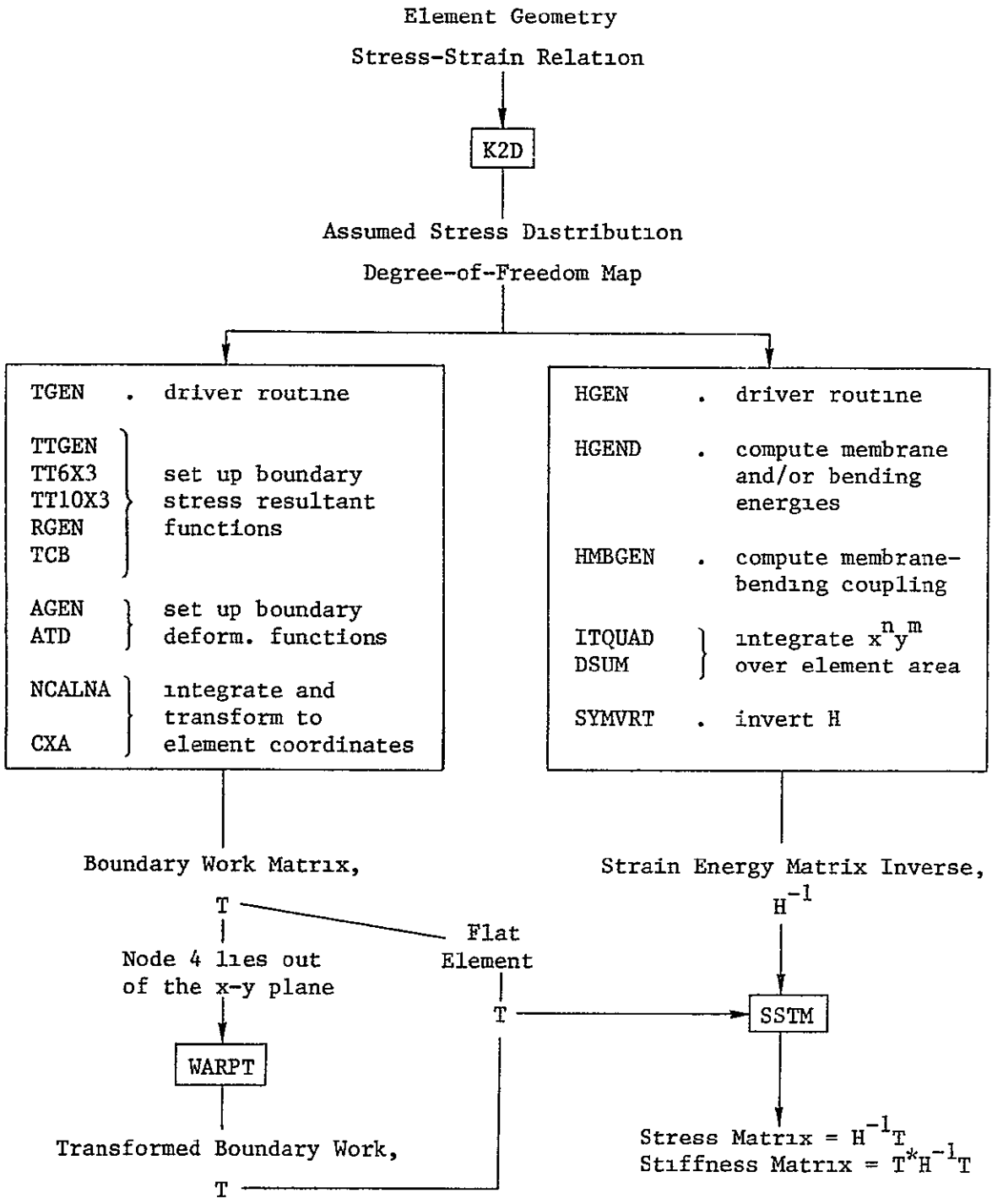
3. Form the element stress matrix = $H^{-1}T$ and the element stiffness matrix = $T^* H^{-1} T$.

---

* Ref. 1. Pian, T.H.H., "Derivation of Element Stiffness Matrices by Assumed Stress Distributions, AIAA J. 2, 1333-1336 (1964).

Ref. 2. SPAR Reference Manual, Vol. II, Section A.

The K2D-series of subroutines are organized as shown below.



The following inputs are supplied through the K2D calling sequence:

A = Upper triangular portion (stored by column) of the element stress-strain relation. The following equations illustrate the required A configurations:

Membrane Elements

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_4 \\ & a_3 & a_5 \\ \text{sym.} & & a_6 \end{bmatrix} \begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} \tag{1}$$

Plate Bending Elements

$$\begin{bmatrix} \gamma_x \\ \gamma_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_4 \\ & a_3 & a_5 \\ \text{sym.} & & a_6 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} \tag{2}$$

Uncoupled Membrane + Bending

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \gamma_x \\ \gamma_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_4 & & & \\ & a_3 & a_5 & \text{Zero} & & \\ & & a_6 & & & \\ & & & a_7 & a_8 & a_{10} \\ & & & & a_9 & a_{11} \\ \text{sym.} & & & & & a_{12} \end{bmatrix} \begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} \tag{3}$$

Coupled Membrane + Bending

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \gamma_x \\ \gamma_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_4 & a_7 & a_{11} & a_{16} \\ & a_3 & a_5 & a_8 & a_{12} & a_{17} \\ & & a_6 & a_9 & a_{13} & a_{18} \\ & & & a_{10} & a_{14} & a_{19} \\ & & & & a_{15} & a_{20} \\ \text{sym.} & & & & & a_{21} \end{bmatrix} \begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} \tag{4}$$

Shear Panels

$$N_{xy} = a_1 \varepsilon_{xy} \quad (5)$$

XX(2, NSIDES), XX(I,J) = direction-I position of node J, relative to the element reference frame

NSIDES, either 3 or 4

NBM = number of assumed membrane stress resultant coefficients (maximum = 10)

NBB = number of assumed bending stress resultant coefficients (maximum = 15)

ISHELL = 0, no membrane-bending coupling
≠ 0, membrane-bending coupling in A

X34 = out-of-plane Z location of node 4

IX34 = 0, no out-of-plane transformation
> 0, transform T to account for non-zero X34

The following arrays are returned through the K2D calling sequence:

S = upper triangular portion (stored by column) of the NQT x NQT element stiffness matrix,

STM (NBT,NQT) = element stress matrix, NBT = NBM + NBB, and

NQT = order of the stiffness matrix.

The following error conditions are returned through the common block ERRK2D (K2D line number 10).

AERR	NERR	Meaning
4HISNG	n	H matrix singularity, row n
4HNEGD	n	n negative diagonal terms encountered computing H^{-1}
4HCORE	n	n additional core locations required in array W
Blank	0	no error

The procedure incorporated in K2D is outlined below.

Step A. (K2D line numbers 20 through 31)

Set up the assumed membrane stress resultant distribution via data statements. The arrays IBM, BM, IXYM, and XYM are used to characterize the equation below.

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} = \begin{bmatrix} b_3 & b_6 & b_4 & b_{10} & b_8 & 0 \\ b_2 & b_5 & b_7 & b_9 & b_{10} & 0 \\ b_1 & -b_7 & -b_6 & 0 & 0 & -2b_{10} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ y^2 \\ xy \end{bmatrix} \qquad (6)$$

For the preceding equation, IBM and BM characterize the matrix of b's as

$$IBM = \begin{bmatrix} 3 & 6 & 4 & 10 & 8 & 0 \\ 2 & 5 & 7 & 9 & 10 & 0 \\ 1 & 7 & 6 & 0 & 0 & 10 \end{bmatrix}, \text{ and}$$

$$BM = \begin{bmatrix} 1. & 1. & 1. & 1. & 1. & 0. \\ 1. & 1. & 1. & 1. & 1. & 0. \\ 1. & -1. & -1. & 0. & 0. & -2. \end{bmatrix}.$$

XYM and IXYM characterize the vector of x's and y's. For the J^{th} term, IXYM (1,J) is the exponent of x, IXYM (2,J) is the exponent of y, and XYM (J) is the coefficient. For example:

Vector		IXYM(1,J)	IXYM(2,J)	XYM(J)
1		0	0	1.
x		1	0	1.
y	$\Rightarrow$	0	1	1.
x^2		2	0	1.
y^2		0	2	1.
xy		1	1	1.

If P(I) = $I^{\underline{th}}$ membrane stress, resultant
(i.e. P(1) = N_x, P(2) = N_y, P(3) = N_{xy}),

BM(I,J) = C,

IBM(I,J) = k,

XYM(I) = Z,

IXYM(1,J) = n, and

IXYM(2,J) = m, then

$$P(I) = P(I) + b_k * C\ Z\ x^n\ y^m, \text{ for } J = 1, 6. \tag{7}$$

For current applications, Z=1.0 for all values of J. This is evident from the dataed array XYM. Provisions for two constants, C and Z, are included for possible future applications.

Current dimensions permit a maximum of 10 membrane stress resultant coefficients. The input variable NBM controls the selection of 10 or less. Note that

NBM = 1 = pure shear stress distribution, used for E44 elements, and

NBM = 3 = constant stress distribution, used for E31 elements.

Step B. (K2D line numbers 33 through 48)

Set up the assumed plate bending stress resultant distributing via data statements. Similar to the membrane representation, BB, IBB, XYB, and IXYB characterize the equation below.

$$\begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} b_1 & 0 & b_6 & 0 & b_4 & 0 & b_{14} & b_{12} & b_{10} & 0 \\ b_2 & 0 & b_5 & 0 & b_7 & 0 & b_{13} & b_{15} & b_{11} & 0 \\ b_3 & 0 & b_9 & 0 & b_8 & 0 & 0 & 0 & b_{14} & b_{15} \\ b_6 & -b_8 & b_{14} & -b_{15} & b_{10} & 0 & 0 & 0 & 0 & 0 \\ b_7 & -b_9 & b_{11} & 0 & b_{15} & -b_{14} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ x \\ x \\ y \\ y \\ x^2 \\ y^2 \\ xy \\ xy \end{bmatrix} \tag{8}$$

Step C. (K2D line number 50)

Set up degree-of-freedom (DOF) indicators

NQM (1) = 3 = 3-node membrane DOF's

NQM (2) = 5 = 4-node membrane DOF's

NQB (1) = 6 = 3-node plate element DOF's

NQB (2) = 9 = 4-node plate element DOF's

Step D. (K2D line numbers 52 through 61)

Set up MAPM(4,5) = membrane DOF map and MAPB(4,5) = plate bending DOF map.

MAPM(I,J) = membrane degree-of-freedom associated with direction-J motion of node I.

MAPB(I,J) = bending degree-of-freedom associated with direction-J motion of node I.

J = 1, 2, 3 = displacement in direction-J

J = 4, 5 = rotation about axis J-3.

Step E. (K2D line numbers 62 through 64)

Initialize error indicators, AERR and NERR.

IWARP = 0 eliminates the possibility of activating alternative boundary work calculations concerning warped boundaries. All warped element considerations are handled by the routine WARPT.

Step F. (K2D line numbers 66 through 71)

Set up the nodal intrinsic position coordinate array, X(2,4). For 3-node elements:

X(I,4) = X(I,3), I= 1,2.

Step G. (K2D line numbers 72 through 99)

Set up the following quantities:

NBOTH = 0, membrane or bending alone
NBOTH = 1, membrane plus bending
NBT = total number of assumed stress resultant coefficients

NXM = order of the vector XYM
NXB = order of the vector XYB
NQT = number of element degrees of freedom
MAPQ(4,5) = element degree-of-freedom map
MAPQ(I,J) = element DOF associated with direction-J motion of node I.

Step H. (K2D line numbers 107 and 108)

Compute the element strain energy matrix inverse, H^{-1}, by calling HGEN. H^{-1} is stored in the array W beginning at location IH.

Step I. (K2D line numbers 111 through 117)

Set up NI(4) and NJ(4). Side K of the element is bounded by nodes NI(K) and NJ(K). The following table summarizes NI and NJ.

	3-Node Elements		4-Node Elements	
K	NI(K)	NJ(K)	NI(K)	NJ(K)
1	1	3	1	4
2	3	2	4	3
3	2	1	3	2
4	-	-	2	1

Step J. (K2D line number 120)

Compute the element boundary work matrix, T, by calling TGEN. T is stored in the array W beginning at location IT.

Step K. (K2D line number 123)

Modify T to account for an out-of-plane node 4 by calling WARPT.

Step L. (K2D line number 126)

Call SSTM to compute the intrinsic stress and stiffness matrices.

ELEMENT STRAIN ENERGY

For a coupled membrane - plate bending finite element formulation, the stress-strain relation for the cross-section is written as

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \gamma_x \\ \gamma_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_2^* & A_3 \end{bmatrix} \begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix}, \text{ or}$$

$$\begin{bmatrix} E_m \\ E_b \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_2^* & A_3 \end{bmatrix} \begin{bmatrix} F_m \\ F_b \end{bmatrix} . \qquad (9)$$

The assumed stress resultant distributions for the element are

$$F_m = P_m B_m, \text{ and}$$

$$F_b = P_b B_b, \text{ where} \qquad (10)$$

$B_m = (b_1\ b_2\ \ldots b_{NBM})^*$ = vector of membrane stress resultant coefficients

$B_b = (b_1\ b_2\ \ldots b_{NBB})^*$ = vector of bending stress resultant coefficients.

NBM and NBB represent the number of membrane and bending stress resultant coefficients assumed for the formulation. All elements of P_m and P_b are of the general form $x^n y^m$.

The element strain energy is expressed in terms of the following area integral

$$V = 1/2 \int_{\text{Area}} \begin{bmatrix} F_m^* & F_b^* \end{bmatrix} \begin{bmatrix} A_1 & A_2 \\ A_2^* & A_3 \end{bmatrix} \begin{bmatrix} F_m \\ F_b \end{bmatrix} \text{ dArea, or}$$

$$V = 1/2 \begin{bmatrix} B_m^* & B_b^* \end{bmatrix} \begin{bmatrix} H_{mm} & H_{mb} \\ H_{mb}^* & H_{bb} \end{bmatrix} \begin{bmatrix} B_m \\ B_b \end{bmatrix} \text{, where}$$

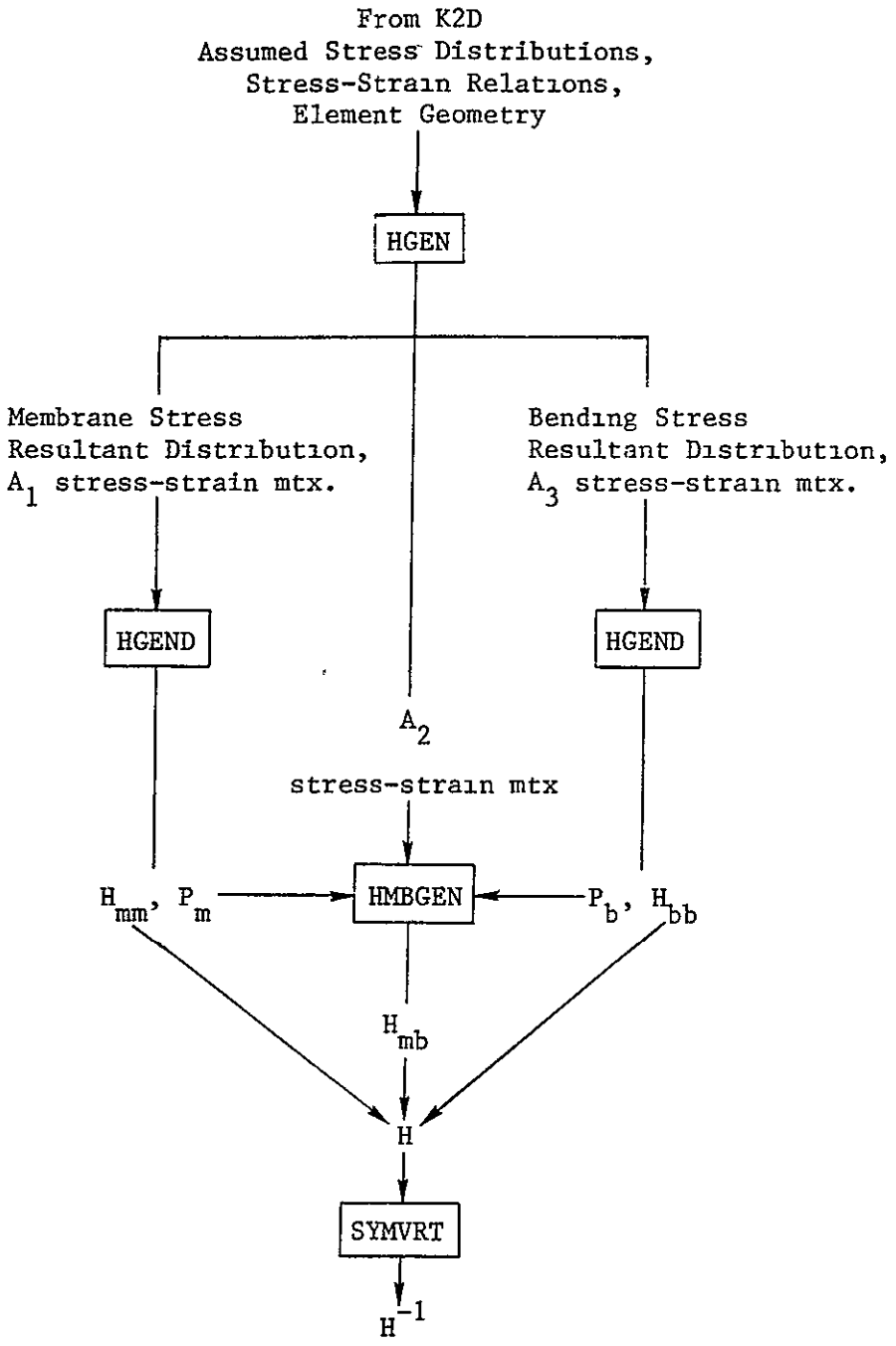
$$H_{mm} = \int_{\text{Area}} \begin{bmatrix} P_m^* & A_1 & P_m \end{bmatrix} \text{ dArea,} \tag{11}$$

$$H_{mb} = \int_{\text{Area}} \begin{bmatrix} P_m^* & A_2 & P_b \end{bmatrix} \text{ dArea, and} \tag{12}$$

$$H_{bb} = \int_{\text{Area}} \begin{bmatrix} P_b^* & A_3 & P_b \end{bmatrix} \text{ dArea.} \tag{13}$$

H_{mm}, H_{mb} and H_{bb} are submatrices of the desired strain energy matrix H.

For a membrane-plate bending coupled element, H^{-1} is computed according to the following chart.



For a pure membrane, plate bending, or uncoupled element formulation only applicable portions of the above chart are followed.

HGEN (PARENT ROUTINE = K2D)

The following inputs are supplied through the HGEN calling sequence:

NBM, NBB = number of assumed membrane and bending stress resultant coefficients

X(2,4) = nodal position coordinates

AA = upper triangular portion (stored by column) of the stress-strain matrix

XYM, IXYM, XYB, IXYB, IBM, BM, IBB, BB = arrays characterizing the assumed membrane and bending stress resultant distributions (see K2D)

NXM, NXB = order of the vectors XYM and XYB

ISHELL = 0, no membrane-bending coupling in AA
≠ 0, membrane-bending coupling in AA

NBT = NBM + NBB

HGEN directs the computation of the element strain energy matrix inverse H(NBT,NBT). H is returned through the HGEN calling sequence.

The procedure incorporated in HCEN is outlined below.

Step A. (HGEN line number 8)

Set up via the data statement:

NC =3, order of the A_1, A_2, and A_3 stress-strain matrices

NSM =3, number of rows in IBM and BM

NSB =5, number of rows in IBB and BB

Step B. (HGEN line numbers 9 and 10)

Initialize the IQT array to 0. In the common block /CK204/

$$QT(I,J) = \int_{Area} x^{(I-1)} \, y^{(J-1)} \, dArea.$$

If IQT(I,J) = 0, QT(I,J) is not computed.
$\neq$ 0, QT(I,J) is computed.

Step C. (HGEN line number 11 through 13)

XX(I) and YY(I) are the x and y positions relative to the element reference frame of node I.

Step D. (HGEN line numbers 15 through 17)

Initialize H to zero.

Step E. (HGEN line numbers 19 through 49)

Set up A(3,3,3), the three stress-strain matrices stored in rectangular fashion. Entries to A are extracted from the input array AA. See Eq. 9.

E1. (HGEN line number 25)

For NBM=1 a shear panel formulation is implemented, and the stress-strain relation consists of a single term as shown in the following equation:

$$N_{xy} = A(1) \, \epsilon_{xy}$$

E2. (HGEN line numbers 27 through 33)

A(I,J,1) = membrane stress-strain matrix

E3. (HGEN line numbers 34 through 42)

A(I,J,3) = bending stress-strain matrix

E4. (HGEN line numbers 43 through 48)

A(I,J,2) = coupling stress-strain matrix

Step F. (HGEN line numbers 53 through 59)

Compute the membrane contribution to H by calling HGEND. Locations within the temporary array W are assigned for storing P_m.

Step G. (HGEN line numbers 61 through 67)

Compute the bending contribution to H by calling HGEND. Locations within the temporary array W are assigned for storing P_b.

Step H. (HGEN line numbers 69 and 70)

Compute the membrane-bending coupling contribution to H by calling HMBGEN.

Step I. (HGEN line number 73)

Invert the H matrix by calling SYMVRT.

HGEND (PARENT ROUTINE = HGEN)

The following inputs are supplied through the HGEND calling sequence:

NB = number of assumed stress coefficients.
For membrane energy, NB = NBM.
For plate-bending energy, NB = NBB.

INCB = stress coefficient increment.
For membrane energy, INCB = 0.
For plate-bending energy, INCB = NBM.

NBT = total number of assumed stress coefficients = NBM + NBB.

C(NC,NC) = stress-strain matrix

XY(NX), IXY(NX)
IB(NS,NX), B(NS,NX) } define the assumed stress distribution (see K2D)

INCC = increment for locating the beginning stress resultant row number within B and IB. INCC = 0 for current applications.

The following arrays are returned through the HGEND calling sequence:

IPX
IPY
P } These arrays define the assumed stress distribution in a configuration that is convenient for computing H terms.

H(NBT,NBT). Contributions to the element strain energy are added to this array.

The procedure incorporated in HGEND is outlined below:

Step A. (HGEND line numbers 11 through 26)

The input stress-resultant distribution matrices (XY, IXY, B, and IB) are rearranged for computing strain energy terms. The input matrices characterize the stress distribution as follows (see K2D discussion):

$$\begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ S_{NS} \end{bmatrix} = \begin{bmatrix} b_i \cdots & \\ b_j \cdots & \\ \cdot & b_{NB} \\ \cdot & \\ \cdot & \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ \cdot \\ \cdot \end{bmatrix} \qquad (14)$$

IB and B — XY and IXY

The following arrangement is used for computing H, Eq. 10.

$$\begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ S_{NS} \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1,NB} \\ P_{21} & P_{22} & \cdots & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ P_{NS,1} & & \cdots & P_{NS,NB} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_{NB} \end{bmatrix} \qquad (15)$$

P, IPX, and IPY

In the above equation b_i is the $i^{\underline{th}}$ stress coefficient and P_{ij} is of the general form $a\, x^n\, y^m$.

The terms of P, IPX, and IPY are extracted from B, IB, XY, and IXY. If $P_{ij} = a\, x^n\, y^m$, then

$$P(I,J) = a$$
$$IPX(I,J) = n$$
$$IPY(I,J) = m.$$

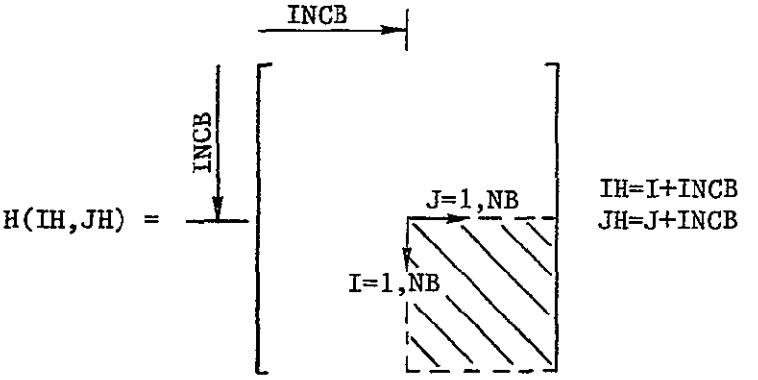
Step B. (HGEND line numbers 28 through 47)

Compute contributions to H of the form

$$\int_{Area} P^* CP \; dArea, \text{ see Eqs. 11 and 13.}$$

B1. (HGEND line numbers 29 through 31)

IH and JH are the row/column designation within H.



B2. (HGEND line numbers 34 through 44)
A typical term in the integral is

$$H(IH,JH) = H(IH,JH) + PC \int_{Area} x^{MX}\, y^{MY}\; dArea \qquad (16)$$

If $\int_{Area} x^{MX}\, y^{MY}\, dArea$ has not been previously computed, ITQUAD is called. The common block /CK2D1/ is used to transfer the necessary data to and from ITQUAD.

HMBGEN (PARENT ROUTINE = HGEN)

The following inputs are supplied through the HMBGEN calling sequence:

NBM and NBB = number of assumed membrane and bending stress-resultant coefficients

C(NCM,NCB) = matrix relating membrane strains to bending stress resultants

IPXM, IPYM, PM } arrays characterizing the assumed membrane stress-resultant distribution, see HGEND

IPXB, IPYB, PB } arrays characterizing the assumed bending stress-resultant distribution

NBT = NBM + NBB = order of the H matrix

Contributions to the element strain energy associated with coupled membrane-bending action are computed by HMBGEN and inserted in H(NBT,NBT).

The procedure incorporated by HMBGEN is outlined below:

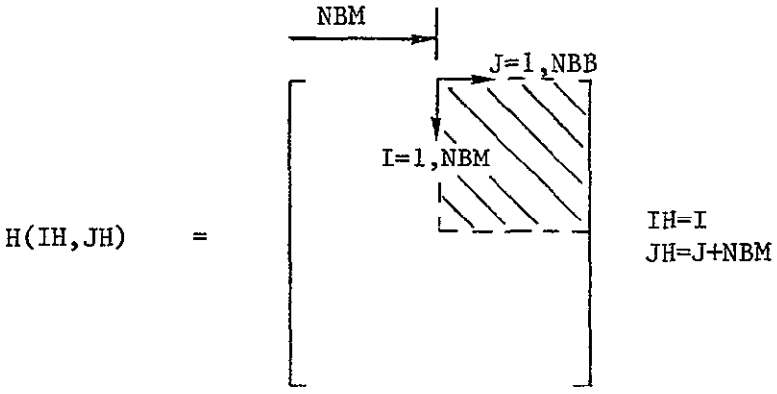
Step A. (HMBGEN line numbers 12 through 30)

Compute contributions to H of the form

$$\int_{\text{Area}} P_m^* \, C \, P_b \, d\text{Area}, \text{ see Eq. 12.}$$

A1. (HMBGEN line numbers 13 through 15)

IH and JH are the row/column designation within H.



A2. (HMBGEN line numbers 18 through 28)

A typical term in the integral is

$$H(IH,JH) = H(IH,JH) + PC \int_{Area} x^{NX}\, y^{MY}\, dArea \qquad (17)$$

As in HGEND, the value of the integral is taken from QX of the common block /CK2D4/

SYMVRT (PARENT ROUTINE = HGEN)

The following inputs are supplied through the SYMVRT calling sequence:

A(N,N)	=	symmetric matrix
NHD	=	dimension of A. The maximum for NHD is 30 due to current dimensions of the temporary arrays B and T.

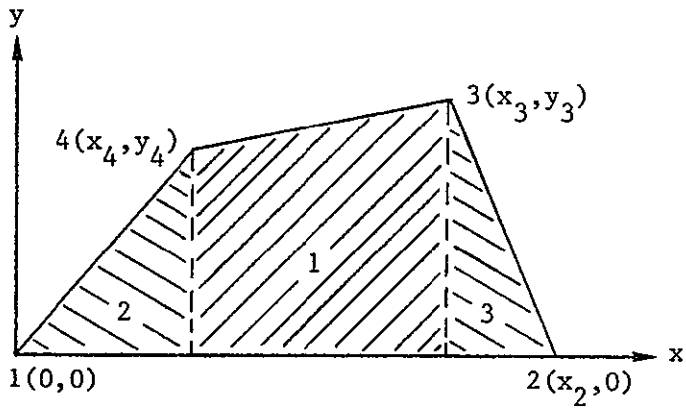
SYMVRT computes A^{-1} and stores the result in the space occupied by the original A matrix.

Error conditions encountered during inversion are returned through the common block /ERRK2D/. If AERR = ISNG, NERR contains the row number of A at which a singularity is encountered. This constitutes an exit condition; the matrix inversion is terminated. If AERR = NEGD, NERR contains the number of negative diagonal terms encountered during inversion. This indicates that A is not positive definite.

QUADRILATERAL AREA INTEGRATOR

A procedure is developed below for integrating a function of the general form $x^n y^m$ over an arbitrarily shaped quadrilateral surface.

The quadrilateral area is illustrated below.



The quadrilateral integral is computed as the summation three partial surface integrals.

$$\int_{Area} x^n y^m \, dArea = A_1 + A_2 + A_3 \tag{18}$$

$$A_1 = \int_{x_4}^{x_3} \int_{0}^{\bar{y}} x^n y^m \, dx \, dy \quad ; \quad \bar{y} = y_4 + \frac{y_3 - y_4}{x_3 - x_4} (x - x_4) \tag{19}$$

$$A_2 = \int_{0}^{x_4} \int_{0}^{\bar{y}} x^n y^m \, dx \, dy \quad ; \quad \bar{y} = \frac{y_4}{x_4} x \tag{20}$$

$$A_3 = \int_{x_3}^{x_2} \int_{0}^{\bar{y}} x^n y^m \, dx \, dy \quad ; \quad \bar{y} = y_3 - \frac{y_3}{x_2 - x_3} (x - x_3) \tag{21}$$

A_1 and A_3 integrals are transformed to relative coordinates, so that

where $\bar{x} = x - x_4$

$$A_1 = \int_0^{(x_3-x_4)} \int_0^{\bar{y}} (\bar{x} + x_4)^n \, y^m \, d\bar{x} \, dy \quad ; \quad \bar{y} = y_4 + \frac{y_3 - y_4}{x_3 - x_4} \bar{x} \, , \qquad (22)$$

where $\bar{x} = x - x_2$

$$A_3 = - \int_0^{(x_3-x_2)} \int_0^{\bar{y}} (x_2 + \bar{x})^n \, y^m \, d\bar{x} \, dy \quad ; \quad \bar{y} = \frac{y_3}{x_3 - x_2} \bar{x} \qquad (23)$$

The general form of A_1 and A_3 is

$$A = \int_0^{T} \int_0^{\bar{y}} (x_L + \bar{x})^n \, y^m \, d\bar{x} \, dy \quad ; \quad \bar{y} = a + b\bar{x}. \qquad (24)$$

Integrating

$$A = \frac{1}{m + 1} \int_0^{T} (x_L + \bar{x})^n \, (a + b\bar{x})^{m+1} \, d\bar{x}. \qquad (25)$$

In the preceding expression

$$(x_L + \bar{x})^n = d_1 \bar{x}^n + d_2 \bar{x}^{n-1} + \ldots d_{n+1}, \text{ and}$$

$$(a + b\bar{x})^{m+1} = c_1 \bar{x}^{m+1} + c_2 \bar{x}^m + \ldots c_{m+2}. \tag{26}$$

A is now rewritten as

$$A = \frac{1}{m+1} \int_0^T \left[d_1 c_1 \bar{x}^{(n+m+1)} + (d_1 c_2 + d_2 c_1) \bar{x}^{n+m} + \ldots d_{n+1}\, c_{m+2} \right] d\bar{x}. \tag{27}$$

A_2 and A_3 are computed according to the above equation. A_2 is evaluated directly as

$$A_2 = \frac{x_4^{n+1}\, y_4^{m+1}}{(n+m+2)\ (m+1)} \tag{28}$$

The routines used to implement this quadrilateral integration procedure are ITQUAD and DSUM.

ITQUAD (PARENT ROUTINES = HGEND and HMBGEN)

The following inputs are supplied through the labeled common block /CK2D1/:

X(4), Y(4) } X(I) and Y(I) are the x and y locations of node I

N = exponent of x

M = exponent of y

ITQUAD directs the computation of

$$AXNYM = \int_{Area} x^N y^M \, dArea.$$

AXNYM is returned in the calling routine through /CK2D1/.

The procedure incorporated in ITQUAD is outlined below:

Step A. (ITQUAD line numbers 7 and 8)

Initialize AR(1), AR(2), and AR(3) to zero.

Step B. (ITQUAD line numbers 9 through 49)

Integrate over zone 1; compute AR(1)

B1. (ITQUAD line numbers 10 through 13)

XL is a nominal element dimension against which other element dimensions are compared to determine whether or not they should be considered zero.

If D/XL < ZERO, D = 0.

B2. (ITQUAD line number 14)

If $x_3 - x_4 = 0$, AR(1) = 0. This convention is used for integrating over triangular surfaces.

B3. (ITQUAD line numbers 15 and 16)

Set up $\bar{y} = A + B\,\bar{x}$

B4. (ITQUAD line numbers 20 through 22)

For B = 0, $y^{M+1} = A^{M+1}$

B5. (ITQUAD line numbers 25 through 33)

For B > 0, $y^{M+1} = C(1)\,\bar{x}^{M+1} + C(2)\,\bar{x}^{M} + \ldots\ C(M+2)$

B6. (ITQUAD line numbers 35 through 44)

$$(x_4 + \bar{x})^{N} = CC(1)\,\bar{x}^{N} + CC(2)\,\bar{x}^{N-1} + \ldots\ CC(N+1) \qquad (29)$$

B7. (ITQUAD line numbers 45 through 49)

Compute the AR(1) integral by calling DSUM. Data is transferred to and from DSUM through the common block /CK2D2/.

$$SUM = \int_0^T \left[C(1)\,\bar{x}^{M+1} + \ldots\right]\left[CC(1)\,\bar{x}^{N} + \ldots\right] d\bar{x} \qquad (30)$$

Step C. (ITQUAD line numbers 51 through 54)

Integrate over zone 2; compute AR(2).
If $x_4 = 0$, AR(2) = 0.

Step D. (ITQUAD line numbers 56 through 75)

Integrate over zone 3; compute AR(3).

D1. (ITQUAD line numbers 59 through 68)

$$(x_2 + \bar{x})^N = CC(1)\ \bar{x}^N + CC(2)\ \bar{x}^{N+1} + \ldots\ CC(N+1) \qquad (31)$$

D2. (ITQUAD line numbers 69 through 73)

Call DSUM to compute

$$SUM = \int_0^T C(1)\ \bar{x}^{M+1} \left[CC(1)\ \bar{x}^N + \ldots\ CC(N+2) \right] d\bar{x} \qquad (32)$$

D3. (ITQUAD line number 75)

Compute AR(3)

Step E. (ITQUAD line numbers 77 through 81)

Compute

$$AXNYM = \frac{1}{M+1} \sum_{I=1}^{3} AR(I). \qquad (33)$$

DSUM (PARENT ROUTINE = ITQUAD)

The following inputs are supplied through the common block /CK2D2/

$$\left.\begin{matrix} N \\ B \end{matrix}\right\} \quad f_1 = B(1)\ \bar{x}^{N} + B(2)\ \bar{x}^{N-1} + \ldots\ B(N+1) \tag{34}$$

$$\left.\begin{matrix} M \\ MM \\ C \end{matrix}\right\} \quad f_2 = C(1)\ \bar{x}^{M} + C(2)\ \bar{x}^{M-1} + \ldots\ C(MM)\ \bar{x}^{M+1-MM} \tag{35}$$

T = integrating limit

DSUM evaluates the following integral and returns the result through /CK2D2/.

$$SUM = \int_0^T f_1\ f_2\ d\bar{x} \tag{36}$$

substituting for f_1 and f_2 and expanding the product yields

$$\begin{aligned} SUM = \int_0^T \Big\{ & B(1)\ C(1)\ \bar{x}^{N+M} + \Big[B(1)\ C(2) + B(2)\ C(1)\Big]\bar{x}^{N+M-1} \\ & + \Big[B(1)\ C(3) + B(2)\ C(2) + B(3)\ C(1)\Big]\ \bar{x}^{N+M-2} \\ & + \ldots + B(N+1)\ C(MM)\ \bar{x}^{M+1-MM} \Big\}\ d\bar{x} \end{aligned} \tag{37}$$

SUM is evaluated by taking each term of the above summation and integrating from $\bar{x} = 0$ to $\bar{x} = T$. A typical term of the sum is

$$D = \int_0^T \left[\sum_{I=IB}^{N1} B(I) * C(I)\ \bar{x}^{IE} \right] d\bar{x}, \text{ where} \tag{38}$$

$$J = N + M - 1 - IE + 2,\ 1 \le J \le MM.$$

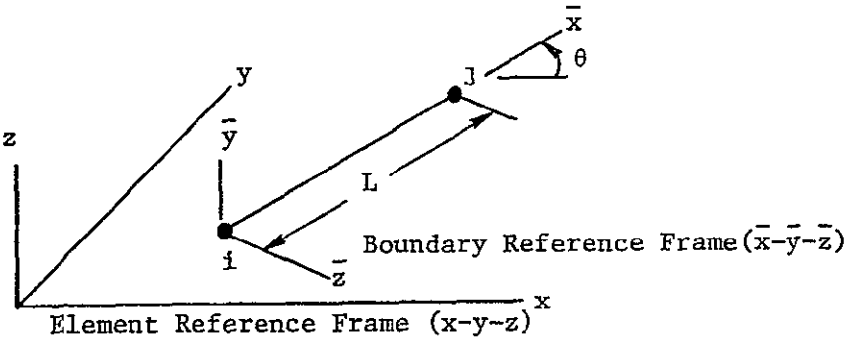
BOUNDARY WORK

The work performed by element stress resultants acting through boundary deformations is computed as

$$W = \sum_{n=1}^{nsides} Wn,$$

where Wn is the work performed along the n^{th} element boundary.

Associated with each element boundary is a boundary reference frame, $\bar{x} - \bar{y} - \bar{z}$, as shown below for boundary n, connecting nodes i and j.

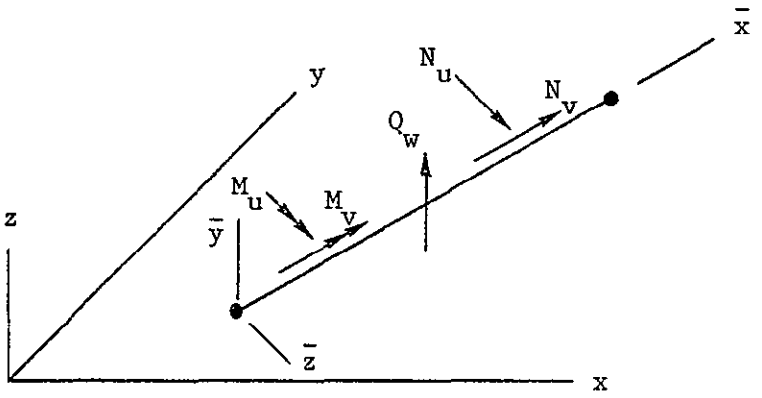


$\bar{y}$ is parallel with z; $\bar{x}$ extends from i through J, and $\bar{z}$ is directed toward the element interior. Points lying along i-J are located by $x = t_1 + \bar{x}\cos\theta$ and $y = t_3 + \bar{x}\sin\theta$.

Functions characterizing the deformation of boundary n are

u ($\bar{x}$) = displacement in direction-$\bar{z}$,
v ($\bar{x}$) = displacement in direction-$\bar{x}$
w ($\bar{x}$) = displacement in direction-$\bar{y}$,
ϕ ($\bar{x}$) = rotation about axis $\bar{z}$, and
θ ($\bar{x}$) = rotation about axis $\bar{x}$.

The element stress resultants acting along boundary n of a coupled membrane plate-bending element are illustrated below.



The boundary work performed along boundary n is

$$Wn = \int_0^L (N_u\, u + N_v\, v + Q_w\, w + M_u\, \phi + M_v\, \Theta)\, d\bar{x}.$$

where $P_1 = N_u$, $P_2 = N_v$, $P_3 = M_u$, $P_4 = M_v$, $P_5 = Q_w$,

$U_1 = u$, $U_2 = v$, $U_3 = \phi$, $U_4 = \Theta$, and $U_5 = w$, the above expression is written as

$$Wn = \sum_{k=1}^{5} \int_0^L P_k\, U_k\, d\bar{x}. \qquad (39)$$

For a pure membrane formulation k=1, 2. For pure plate-bending k=3, 4, 5. For coupled membrane plus bending k=1 through 5.

Each boundary stress resultant, P_k, and each boundary deformation function, U_k, is represented by a polynomial function in $\bar{x}$,

as shown below

$$P_k = (C_1\ C_2\ \ldots\ C_{NC}) \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \\ \cdot \\ \cdot \\ \bar{x}^{NC-1} \end{bmatrix} = C^* X_c\ , \qquad (40)$$

$$U_k = (1\ \bar{x}\ \ldots\ \bar{x}^{NA-1}) \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_{NA} \end{bmatrix} = X_a^* a. \qquad (41)$$

The boundary work performed along boundary n may now be expressed as

$$W_n = \sum_{k=1}^{5} C^* \underbrace{\left[\int_0^L X_c\ X_a^*\ d\bar{x} \right]}_{\searrow} a \qquad (42)$$

$$= \sum_{k=1}^{5} C^*\ X_{ca}\ a$$

Where B represents the vector of undetermined stress-resultant coefficients and Q represents the vector of intrinsic nodal motions, the following transformations are performed:

$$C = T_{cb}\ B \text{ and } a = T_{aq}\ Q.$$

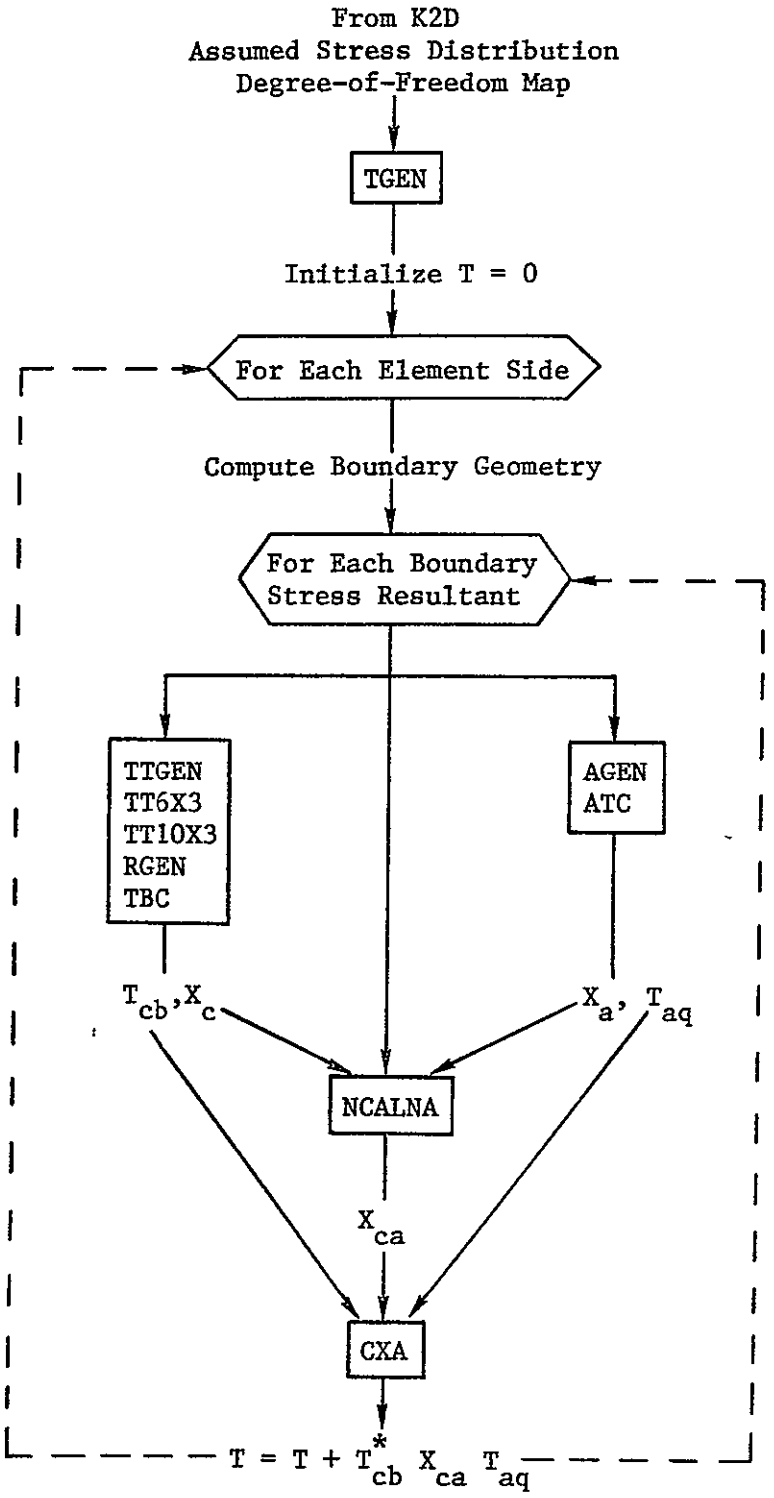
$$Wn = \sum_{k=1}^{5} B^*\ T_{cb}^*\ X_{ca}\ T_{aq}\ Q. \qquad (43)$$

Summing over all sides

$$W = B^{*} TQ, \text{ where } T = \sum_{n=1}^{nsides} \sum_{k=1}^{5} T^{*}_{cb} X_{ca} T_{aq}. \qquad (44)$$

T is the desired boundary work matrix.

T is computed according to the following chart.



TGEN (PARENT ROUTINE = K2D)

The following inputs are supplied through the TGEN calling sequence.

IBM, BM = element membrane stress-resultant distribution matrices (see K2D)

NBM = number of assumed membrane stress-resultant coefficients

IBB, BB = element plate-bending stress-resultant distribution matrices (see K2D)

NBB = number of assumed bending stress-resultant coefficients

MAPQ = element degree-of-freedom map

NBT = NBM + NBB

NQT = number of element intrinsic degrees of freedom

X, X(I,J) = direction-I position relative to the element reference frame of node J

NI, NJ NI(n) = i, NJ(n) = j, the n^{th} element boundary connects element nodes i and j.

TGEN directs the computation of the boundary work matrix T(NBT,NQT).

The procedure incorporated in TGEN is outlined below:

Step A. (TGEN line numbers 13 and 14)

Set up MAPU(3,5) via data statement. For MAPU(1,J) = K, the K^{th} boundary motion function is associated with the J^{th} boundary stress resultant for computing boundary work. MAPU(2,J) and MAPU(3,J) pertain to warped boundary calculations and are not currently used.
(J = IP, pg 36 and K = IU, pg 47)

Step B. (TGEN line numbers 18 through 20)

Initialize the T matrix to zero

Steps C through G are repeated for each boundary.

Step C. (TGEN line numbers 24 through 37)

Set up the boundary geometry as:

T1 = x location of boundary origin
T3 = y location of boundary origin
ZL = boundary length
CS = cos θ, θ = angle between x and $\bar{x}$
SN = sin θ

Steps D through G are repeated for each boundary stress resultant.

Stress Resultant Identification:

IP = 1, N_u } membrane resultants
IP = 2, N_v } active if NBM > 0.

IP = 3, M_u }
IP = 4, M_v } bending resultants
IP = 5, Q_w } active if NBB > 0.

Step D. (TGEN line numbers 40 through 55)

For membrane stress resultants (IP=1 and IP=2) TTGEN is called with a final argument of 1. For bending resultants TTGEN is called with a final argument of 2.

Calls to TTGEN, RGEN, and TCB result in the computation of arrays representing X_c and T_{cb} of Eqs. 40 and 43. NMC represents X_c and C contains T_{cb}.

Step E. (TGEN line numbers 56 through 68)

The boundary deformation function associated with stress resultant IP is set up by AGEN with the final argument IU. IU = MAPU(1,IP). NMA represents X_a (Eq. 41). The transformation matrix T_{aq} (Eq. 43) is stored in a mapped fashion according to arrays A, NQA, NQ1, and IDQ.

Step F. (TGEN line number 70)

Calling NCALNA results in the calculation of

$$X_{ca} = \int_0^{ZL} X_c \, X_a^t \, d\bar{x} \quad \text{(Eq. 42)}.$$

X_{ca} is stored in the array XNCNA.

Step G. (TGEN line number 77)

CXA performs the transformation

$$T_{cb}^{*} \, X_{ca} \, T_{aq} \quad \text{(Eq 43)}.$$

The result is added to the existing T matrix.

TTGEN (PARENT ROUTINE = TGEN)

The following inputs are supplied through the TTGEN calling sequence:

IKIND = 1, membrane stress resultants
= 2, plate-bending stress resultants

T1, T2, T3, T4 : $x = T1 + T2\,\bar{x}$, $y = T3 + T4\,\bar{x}$ — boundary line equation; $\underline{x}$-y = element ref. frame; $\bar{x}$ = boundary ref. frame (45)

The following arrays are returned through the TTGEN calling sequence:

TT = transformation matrix set up by TT6X3 or TT10X3

NMC(1,NC) = representation of X_c (Eq. 40) NMC (2,NC) is not used.

The procedure incorporated in TTGEN is outlined below.

Step A. (TTGEN line numbers 4 through 6)

For membrane stress resultants, TT6X3 is called to generate TT(NX,NC).

Step B. (TTGEN line numbers 8 through 10)

For bending stress resultants, TT10X3 is called to generate TT(NX,NC).

Step C. (TTGEN line numbers 12 through 16)

Set up NMC so that NMC (1,I) = n indicates that $X_c(I) = \bar{x}^n$.

TT6X3 (PARENT ROUTINE = TTGEN)

The following inputs are supplied through the TT6X3 calling sequence.

T1, T2, T3, T4 — boundary line equation, see TTGEN

The transformation matrix T is returned through the TT6X3 calling sequence. Where x and y refer to element reference frame axes and $\bar{x}$ refers to the boundary reference frame axis,

$$\begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ y^2 \\ xy \end{bmatrix} = T(6,3) \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \end{bmatrix}, \text{ or } X_m = T\, X_c \qquad (46)$$

The X_m vector corresponds to the assumed membrane stress-resultant distribution presented in the K2D discussion

TT10X3 (PARENT ROUTINE = TTGEN)

The following inputs are supplied through the TT10X3 calling sequence:

T1, T2, T3, T4 — boundary line equations, see TTGEN

The transformation matrix TT is returned through the TT10X3 calling sequence. Where x and y refer to element reference frame axes and $\bar{x}$ refers to the boundary reference frame axis,

$$\begin{bmatrix} 1 \\ 1 \\ x \\ x \\ y \\ y \\ x^2 \\ y^2 \\ xy \\ xy \end{bmatrix} = TT(10,3) \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \end{bmatrix}, \text{ or } X_b = TT \cdot X_c. \qquad (47)$$

The X_b vector corresponds to the assumed bending stress-resultant distribution presented in the K2D discussion.

RGEN (PARENT ROUTINE = TGEN)

The following inputs are supplied through the RGEN calling sequence.

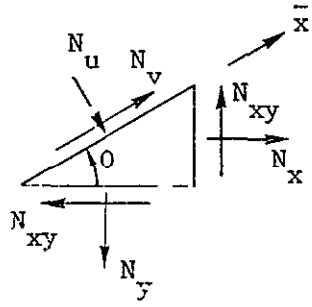
IP = 1, N_u
= 2, N_v
= 3, M_u
= 4, M_v
= 5, Q_w

IP identifies the active boundary stress resultant.

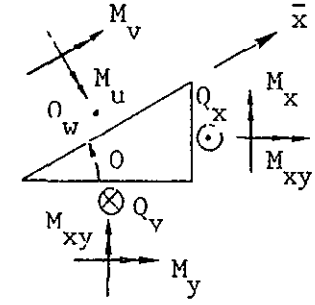
SN = sin θ
CS = cos θ

Define the orientation of the boundary relative to the element ref. frame.

The array R is returned through the RGEN calling sequence. R is used to perform the transformation from element stress resultants to boundary stress resultants. The figures below indicate the transformation matrices that are required. Directions x and y refer to element reference frame directions.



$$\begin{bmatrix} N_u \\ N_v \end{bmatrix} = \begin{bmatrix} -SN^2 & -CS^2 & 2SN\text{-}CS \\ -SN\ CS & SN\cdot CS & (CS^2\text{-}SN^2) \end{bmatrix} \begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} \quad (48)$$



$$\begin{bmatrix} M_u \\ M_v \end{bmatrix} = \begin{bmatrix} SN\ CS & -SN\ CS & (CS^2\text{-}SN^2) \\ -SN^2 & -CS^2 & -2CS\ SN \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} \quad (49)$$

$$Q = \begin{bmatrix} -SN & CS \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

The R array is computed such that for

$$\underline{IP=1}: \quad N_u = R(3) \begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix}, \quad \underline{IP=2}: \quad N_v = R(3) \begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix},$$

$$\underline{IP=3}: \quad M_u = R(5) \begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix}, \quad \underline{IP=4}: \quad M_v = R(5) \begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix}, \text{ and} \qquad (50\text{-}54)$$

$$\underline{IP=5}: \quad Q_w = R(5) \begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix}.$$

TCB (PARENT ROUTINE = TGEN)

Calculation of the T_{cb} transformation matrix (Eq. 43) is based on the following derivation.

$$P_k = (C_1\ C_2\ \cdots\ C_{NC}) \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \\ \vdots \\ \vdots \end{bmatrix}$$

- a boundary stress resultant in boundary ref. frame coordinates. See Eq 40. (55)

$$\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{NI} \end{bmatrix} = \begin{bmatrix} B_{11}\ b_i & B_{12}\ b_j & \cdots \\ & & \\ B_{NI,1}\ b_k & \cdots & \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ \vdots \\ \vdots \end{bmatrix}$$

- the assumed element stress-resultant distribution (either membrane or bending) in element ref. frame coordinates b_1 = $1^{\underline{th}}$ stress coefficient. See K2D. (56)

$$P_k = (R_1\ R_2\ \cdots\ R_{NI}) \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{NI} \end{bmatrix}$$

- transformation from element stress resultants to boundary stress resultants. See RGEN. (57)

$$\begin{bmatrix} 1 \\ x \\ y \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} Z_{11}\ Z_{12}\ \cdots\ Z_{1,NC} \\ Z_{21} \\ \vdots \\ \vdots \\ Z_{NJ,1} \qquad Z_{NJ,NC} \end{bmatrix} \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \\ \vdots \\ \vdots \end{bmatrix}$$

- transformation from boundary coordinates to element coordinates. See TTGEN. (58)

Substituting Eqs. 58 and 56 into Eq. 57 yields the following

$$P_k = (R_1\ R_2\ \cdots\ R_{NI}) \begin{bmatrix} B_{11}\ b_1 & B_{12}\ b_j & \cdots \\ \vdots & & \\ B_{NI,1}\ b_k & \cdots & \end{bmatrix} \begin{bmatrix} Z_{11}\ Z_{12}\ \cdots\ Z_{1,NC} \\ Z_{21} \\ \vdots \\ \vdots \\ Z_{NJ,1} \qquad Z_{NJ,NC} \end{bmatrix} \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \\ \vdots \\ \vdots \end{bmatrix} \tag{59}$$

$$= (C_1\ C_2\ \cdots\ C_{NC}) \begin{bmatrix} 1 \\ \bar{x} \\ \bar{x}^2 \\ \vdots \\ \vdots \end{bmatrix} .$$

Comparing like terms of the above equation (i.e like powers of $\bar{x}$) yields the desired transformation matrix as:

$$\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{NC} \end{bmatrix} = TCB(NC,NB) \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{NB} \end{bmatrix} . \tag{60}$$

The following inputs are supplied through the TCB calling sequence.

NI, NJ, B(NI,NJ), IB(NI,NJ) — matrices characterizing the assumed stress-resultant distribution. These arrays originate in K2D.

R(NI) = element stress resultant to boundary stress resultant transformation array. R is created in RGEN.

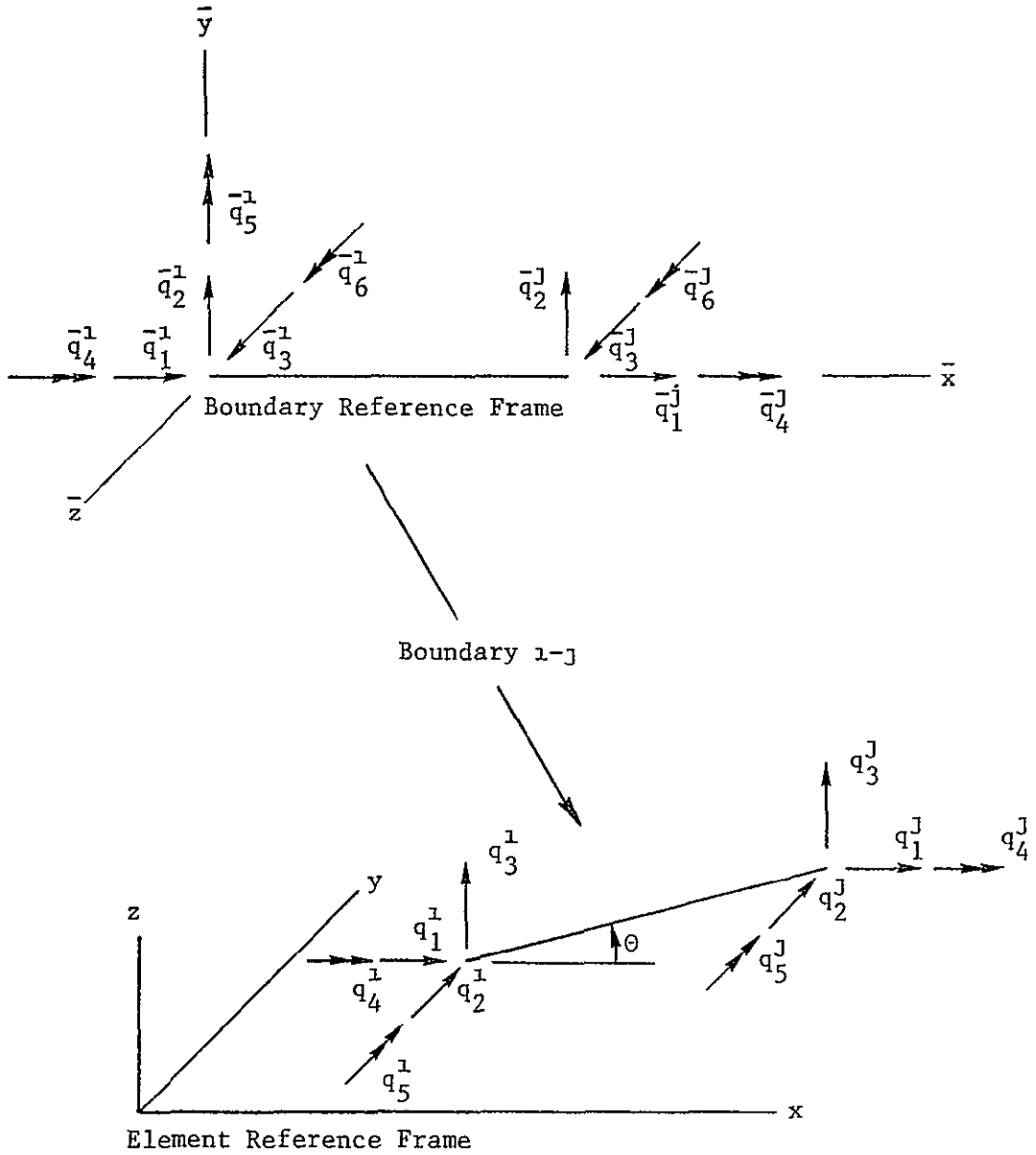
NC, Z(NJ,NC) — transformation from boundary coordinates to element coordinates. Z is created in TTGEN.

NB = number of stress resultant coefficients used in the formulation

The matrix TCB(NC,NB) is computed and returned through the TCB calling sequence

AGEN (PARENT ROUTINE = TGEN)

Displacements and rotations relative to the boundary reference frame and relative to the element reference frame of boundary i-j end points are shown below.



For boundary i-j a deformation function, U_k (Eq. 41), of the following form is assumed:

$$U_k = (1 \ \bar{x} \ \bar{x}^2 \ \ldots) \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{NA} \end{bmatrix} = \text{polynomial with NA forms} = X_a^t \, a \tag{61}$$

The vector of polynomial coefficients is expressed in terms of the motions of nodes i and j, so that

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{NA} \end{bmatrix} = \begin{bmatrix} & \\ AB(1,NA,NQ1B) & AB(2,NA,NQ1B) \\ & \end{bmatrix} \begin{bmatrix} \bar{q}_\ell^i \\ \bar{q}_m^i \\ \bar{q}_\ell^j \\ \bar{q}_m^j \end{bmatrix} \quad \text{NQ1B = 2 in this equation} \tag{62}$$

Transformation from the element reference frame to the boundary reference frame yields

$$\begin{bmatrix} \bar{q}_\ell^h \\ \bar{q}_m^h \end{bmatrix} = \begin{bmatrix} D(NQ1B,NQ1) \end{bmatrix} \begin{bmatrix} q_n^h \\ q_o^h \\ q_p^h \end{bmatrix} \quad \text{NQ1 = 3 in this equation} \tag{63}$$

h = i or j.

For the above equation, the AGEN array IDQ would contain:
IDQ(1) = n, IDQ(2) = o, and IDQ(3) = p.

Performing the transformation indicated by the previous equation yields

$$\begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_{NA} \end{bmatrix} = \begin{bmatrix} A(NA,NQA) \end{bmatrix} \begin{bmatrix} q_n^1 \\ q_o^1 \\ q_p^1 \\ q_n^J \\ q_o^J \\ q_p^J \end{bmatrix}, \text{ where} \qquad (64)$$

$$\begin{bmatrix} A \end{bmatrix} = \left[AB(1,NA,NQ1B) \cdot D \;\middle|\; AB(2,NA,NQ1B) \cdot D \right].$$

The following inputs are supplied through the AGEN calling sequence

ZL = boundary length

SN = $\sin\theta$ } orientation of the boundary
CS = $\cos\theta$ } relative to the element ref. frame

IU = 1, displacement in $\bar{x}$ direction
= 2, displacement in $\bar{y}$ direction
= 3, displacement in $\bar{z}$ direction
= 4, rotation about $\bar{x}$-axis
= 5, rotation about $\bar{y}$-axis
= 6, rotation about $\bar{z}$-axis
= 7, for warped boundary (not currently used)

The following arrays are returned through the AGEN calling sequence:

A(NA,NQA)
IDQ(NQ1)
NA
NQA
NQ1
} the meanings of these arrays are as indicated in the preceding discussion

NMA(1,NA) = representation of X_a
NMA(2,NA) is not currently used

The procedure incorporated in AGEN is outlined below.

Step A. (AGEN line numbers 4 through 106)

Set up NA, NQ1, NQ1B, IDQ, AB, and D according to the input value of IU.

IU = 1 (AGEN line numbers 11 through 20)

U_k = displacement in $\bar{x}$-direction

$= a_1 + a_2 \bar{x}$

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \left[\begin{array}{c|c} 1 & 0 \\ -1/L & 1/L \end{array}\right] \begin{bmatrix} \bar{q}_1^1 \\ \bar{q}_1^J \end{bmatrix} \tag{65}$$

$$\bar{q}_1^h = \begin{bmatrix} CS & SN \end{bmatrix} \begin{bmatrix} q_1^h \\ q_2^h \end{bmatrix} \tag{66}$$

IU = 2 (AGEN line numbers 22 through 42)

U_k = displacement in $\bar{y}$-direction

$= a_1 + a_2 \bar{x} + a_3 \bar{x}^2 + a_4 \bar{x}^3$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/L^2 & -2/L & 3/L^2 & -1/L \\ 2/L^3 & 1/L^2 & -2/L^3 & 1/L^2 \end{array}\right] \begin{bmatrix} \bar{q}_2^1 \\ \bar{q}_6^1 \\ \bar{q}_2^J \\ \bar{q}_6^J \end{bmatrix} \tag{67}$$

$$\begin{bmatrix} \bar{q}_2^h \\ \bar{q}_6^h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & SN & -CS \end{bmatrix} \begin{bmatrix} q_3^h \\ q_4^h \\ q_5^h \end{bmatrix} \tag{68}$$

IU = 3 (AGEN line numbers 44 through 53)

U_k = displacement in $\bar{z}$-direction

$= a_1 + a_2 \bar{x}$

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \left[\begin{array}{c|c} 1 & 0 \\ \hline -1/L & 1/L \end{array}\right] \begin{bmatrix} \bar{q}_3^I \\ \bar{q}_3^J \end{bmatrix} \tag{69}$$

$$\bar{q}_3^h = \begin{bmatrix} SN & -CS \end{bmatrix} \begin{bmatrix} q_1^h \\ q_2^h \end{bmatrix} \tag{70}$$

IU = 4 (AGEN line numbers 55 through 64)

U_k = rotation about the $\bar{x}$-axis

$= a_1 + a_2 \bar{x}$

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \left[\begin{array}{c|c} 1 & 0 \\ \hline -1/L & 1/L \end{array}\right] \begin{bmatrix} \bar{q}_4^i \\ \bar{q}_4^j \end{bmatrix} \tag{71}$$

$$\bar{q}_4^h = \begin{bmatrix} CS & SN \end{bmatrix} \begin{bmatrix} q_4^h \\ q_5^h \end{bmatrix} \tag{72}$$

IU = 5 (AGEN line numbers 66 through 74)

U_k = rotation about the $\bar{y}$-axis

$= a_1$

$$a_1 = \begin{bmatrix} 1/L & -1/L \end{bmatrix} \begin{bmatrix} \bar{q}_3^I \\ \bar{q}_3^J \end{bmatrix} \tag{73}$$

$$\bar{q}_3^h = \begin{bmatrix} SN & -CS \end{bmatrix} \begin{bmatrix} q_1^h \\ q_2^h \end{bmatrix} \tag{74}$$

IU = 6 (AGEN line numbers 76 through 95)

U_k = rotation about the $\bar{z}$-axis

$= a_1 + a_2 \bar{x} + a_3 \bar{x}^2$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ -6/L^2 & -4/L & 6/L^2 & -2/L \\ 6/L^3 & 3/L^2 & -6/L^3 & 3/L^2 \end{array}\right] \begin{bmatrix} \bar{q}_2^1 \\ \bar{q}_6^1 \\ \bar{q}_2^j \\ \bar{q}_6^j \end{bmatrix} \tag{75}$$

$$\begin{bmatrix} \bar{q}_2^h \\ \bar{q}_6^h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & SN & -CS \end{bmatrix} \begin{bmatrix} q_3^h \\ q_4^h \\ q_5^h \end{bmatrix} \tag{76}$$

IU = 7 (AGEN line numbers 98 through 104)

U_k = displacement in the $\bar{y}$-direction for warped membranes = $a + a \bar{x}$

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \left[\begin{array}{c|c} 1 & 0 \\ & \\ -1/L & 1/L \end{array}\right] \begin{bmatrix} \bar{q}_2^1 \\ \bar{q}_2^j \end{bmatrix} \tag{77}$$

$$\bar{q}_2^h = q_3^h \tag{78}$$

Step B. (AGEN line numbers 107 through 111)

Set up NMA(1,I), I = 1,NA. NMA characterizes the vector X_a, so that for

$X_a(J) = \bar{x}^k$,

NMA(1,J) = k.

NMA(1,1) = 0, 1.

NMA(1,2) = 1, $\bar{x}$

⋮

NMA(1,NA) = NA-1, $\bar{x}^{(NA-1)}$

NMA(2,NA) is not currently used.

Step C. (AGEN line numbers 112 and 113)

Call ATD to perform the AB-to-A transformation.

ATD (PARENT ROUTINE = AGEN)

The following inputs are supplied through the ATD calling sequence.

AB(2, NA, NQ1B)
D(NQ1B, NQ1)
NA
NQ1B
NQ1

} The contents of these arrays is discussed under AGEN.

NQA = 2 * NQ1

ATD computes and returns the array A through the calling sequence. A is computed as

$$\left[A(NA,NQA)\right] = \left[AB(1,NA,NQ1B) \cdot D \;\vdots\; AB(2,NA,NQ1B) \cdot D\right]. \qquad (79)$$

NCALNA (PARENT ROUTINE = TGEN)

The following inputs are supplied through the NCALNA calling sequence:

XL = boundary length

NMC, for $X_c(J) = \bar{x}^k$, NMC(1,J) = k
NMC(2,J) is not used

NMA, for $X_a(J) = \bar{x}^k$, NMA(1,J) = k
NMA(2,J) is not used

NC = order of X_c

NA = order of X_a

NALFA, ALFA, SIGN } these arrays pertain to warped element boundaries and are not currently used

NCALNA computes and returns through the calling sequence the array Z. Z is computed in accordance with Eq. 42 as

$$Z(NC,NA) = \int_0^{XL} X_c \, X_a^* \, d\bar{x} \tag{80}$$

The procedure incorporated in NCALNA is outlined below

Step A. (NCALNA line numbers 8 through 13)

$$Z(I,J) = \int_0^{XL} \bar{x}^n \, d\bar{x} = \frac{(XL)^{N+1}}{(N+1)} \text{ , where} \tag{81}$$

N = NMC(1,I) + NMA(1,J)

Since NCALNA is called several times for each element boundary, the integral of $\bar{x}^N$ may be required repeatedly. To avoid duplicate calculation, the common block /CK2D5/ containing IQX(10) and QX(10) is maintained, so that

$$QX(N+1) = \int_0^{XL} \bar{x}^N \, d\bar{x}. \tag{82}$$

If IQX(I) = 0, QX(I) has not been computed.
If IQX(I) ≠ 0, QX(I) has been computed and is present in CK2D5.

Step B. (NCALNA line numbers 15 through 24)

This portion of the code pertains to warped boundary considerations and is not currently used.

CXA (PARENT ROUTINE = TGEN)

The following inputs are supplied through the CXA calling sequence:

C(NC,NBC) = T_{cb} of Eq. 43, from TCB
X(NC,NA) = X_{ca} of Eq. 43, from NCALNA
A(NA,NQA) = T_{aq} of Eq. 43, from AGEN

NA, NQA, NC, NBC } dimensions of the above matrices

NQT = number of element intrinsic degrees of freedom

NBT = number of assumed stress resultant coefficients in the formulation

INCB = row increment

MAPQ(4,5) = degree-of-freedom map

IDQ(NQ1), NQ1 } see AGEN

NODES = 2 for current application

NIJ(1) = i, NIJ(2) = j } this boundary connects nodes i and j

The following arrays are returned through the CXA calling sequence:

$$T(NBT,NQT) = T(NBT,NQT) + C^{*} \times A \tag{83}$$

$$TMP(NBC,NA) = C^{*} \times = \text{temporary array} \tag{84}$$

The procedure incorporated in CXA is outlined below:

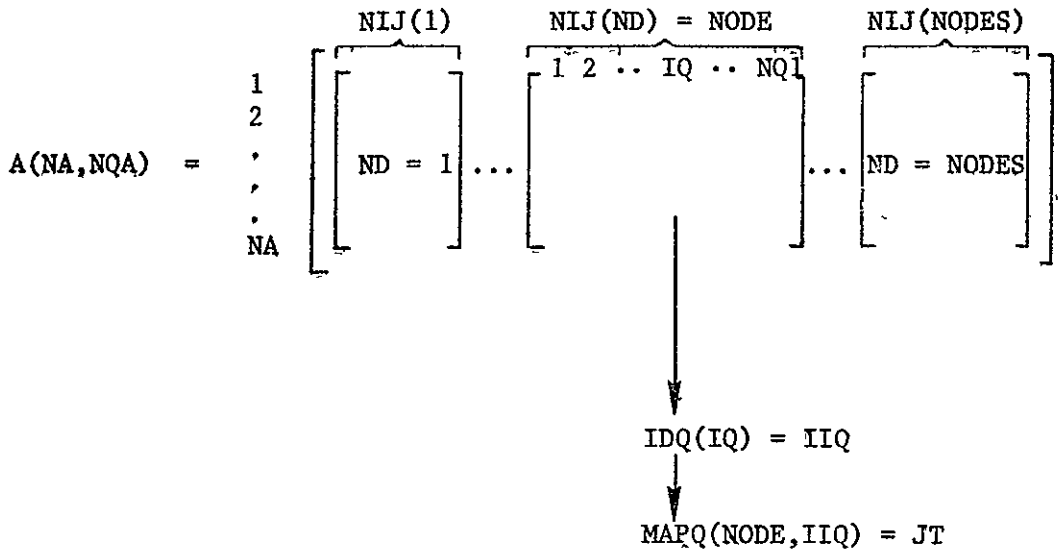
Step A. (CXA line numbers 9 through 13)

Compute TMP(NBC,NA) = C^{*} X

Step B. (CXA line numbers 15 through 27)

Compute T(NBT,NQT) = TMP(NBC,NA) · A

The row location in T of the product TMP·A is controlled by the increment INCB. The column location is controlled by the three maps MAPQ, IDQ, and NIJ.



(85)

The product of TMP with this column of A is added to column JT of T.

Note that NODES = 2 for the current application.

SSTM (PARENT ROUTINE = K2D)

The following inputs are supplied through the SSTM calling sequence:

H(NHD,NHD) = inverse of the element strain energy matrix

T(NHD,NQ) = boundary work matrix

NHD
NQ dimensions of the above matrices

NB = NHD in the current application

The following arrays are returned through the SSTM calling sequence:

STM(NB,NQ) = element stress matrix

S(1) = element stiffness matrix

The following procedure is incorporated in SSTM:

Step A. (SSTM line numbers 3 through 7)

Compute STM = HT (86)

Step B. (SSTM line numbers 8 through 14)

Compute $S = T^* HT = T^* STM$. S is stored by column such that

$$S = \begin{bmatrix} S_1 & S_2 & S_4 & & \\ & S_3 & S_5 & & \\ & & S_6 & \ddots & \\ & & & \ddots & \\ & & & & S_N \end{bmatrix}, \quad N = \frac{NQ\ (NQ+1)}{2} \tag{87}$$

PROGRAM LISTING

The routines comprising the K2D element formulation are listed, in the order shown in the table of contents on this page and the following 18 pages.


```
      SUBROUTINE K2D(A,XX,S,STM,NSIDES,NBM,NBB,ISHELL,NQT,X34,IX34,IERR)
C
      DIMENSION A(1),X(2,4),XX(2,1)
     *,    IBM(3,6),BM(3,6),XYM(6),IXYM(2,6)
     *,    IBB(5,10),BB(5,10),XYB(10),IXYB(2,10)
     *,    MAPM(4,5),MAPB(4,5),MAPQ(4,5),NQM(2),NQB(2),NI(4),NJ(4)
     *,    W(1000)
      COMMON/WARP/ IWARP,NALFA(4),ALFA(5,4),NGAMA(4),GAMA(5,4)
      COMMON/ERRK2D/AERR,NERR
      COMMON/CK2D10/JUMP(10)
C* LABELED COMMON BLOCKS USED WITHIN K2D ROUTINES ARE
C*    /CK2D1/DAT(11)
C*    /CK2D2/DAT(105)
C*    /CK2D3/DAT(3)
C*    /CK2D4/DAT(49)
C*    /CK2D5/DAT(20)
C*    /CK2D10/DAT(10)
C
      DATA ((IBM(I,J),J=1,6),I=1,3),((BM(I,J),J=1,6),I=1,3) /
     *   3,  6,  4, 10,  8,  0,
     *   2,  5,  7,  9, 10,  0,
     *   1,  7,  6,  0,  0, 10,
     *   1., 1., 1., 1., 1., 0.,
     *   1., 1., 1., 1., 1., 0.,
     *   1.,-1.,-1., 0., 0.,-2.   /
C
      DATA ((IXYM(I,J),J=1,6),I=1,2),(XYM(I),I=1,6)  /
     *   0,  1,  0,  2,  0,  1,
     *   0,  0,  1,  0,  2,  1,
     *   1., 1., 1., 1., 1., 1.    /
C
      DATA ((IBB(I,J),J=1,10),I=1,5),((BB(I,J),J=1,10),I=1,5)  /
     *   1,  0,  6,  0,  4,  0, 14, 12, 10,  0,
     *   2,  0,  5,  0,  7,  0, 13, 15, 11,  0,
     *   3,  0,  9,  0,  8,  0,  0,  0, 14, 15,
     *   6,  8, 14, 15, 10,  0,  0,  0,  0,  0,
     *   7,  9, 11,  0, 15, 14,  0,  0,  0,  0,
     *   1., 0., 1., 0., 1., 0., 1., 1., 1., 0.,
     *   1., 0., 1., 0., 1., 0., 1., 1., 1., 0.,
     *   1., 0., 1., 0., 1., 0., 0., 0., 1., 1.,
     *   1.,-1., 1.,-1., 1., 0., 0., 0., 0., 0.,
     *   1.,-1., 1., 0., 1.,-1., 0., 0., 0., 0.   /
C
      DATA ((IXYB(I,J),J=1,10),I=1,2),(XYB(I),I=1,10)  /
     *   0,  0,  1,  1,  0,  0,  2,  0,  1,  1,
     *   0,  0,  0,  0,  1,  1,  0,  2,  1,  1,
```

```
47:     *   1., 1., 1., 1., 1., 1., 1., 1., 1., 1.  /
48:C
49:      DATA (NQM(I),I=1,2),(NQB(I),I=1,2) / 3,5,6,9 /
50:C
51:      DATA ((MAPM(I,J),J=1,5),I=1,4)  /
52:     *   0,  0,  0,  0,  0,
53:     *   1,  0,  0,  0,  0,
54:     *   2,  3,  0,  0,  0,
55:     *   4,  5,  0,  0,  0   /
56:      DATA ((MAPB(I,J),J=1,5),I=1,4)  /
57:     *   0,  0,  0,  0,  0,
58:     *   0,  0,  1,  2,  3,
59:     *   0,  0,  4,  5,  6,
60:     *   0,  0,  7,  8,  9   /
61:      KW=1000
62:      IWARP=0
63:      AERR=4H
64:      NERR=0
65:C
66:      DO 600 I=1,2
67:      DO 600 J=1,3
68:  600 X(I,J)=XX(I,J)
69:      J=NSIDES
70:      X(1,4)=XX(1,NSIDES)
71:      X(2,4)=XX(2,NSIDES)
72:      NBOTH=0
73:      IF(NBM.GT.0.AND.NBB.GT.0) NBOTH=1
74:      NBT=NBM+NBB
75:      NS1=NSIDES-2
76:      NMQ=0
77:      NBQ=0
78:      IF(NBM.GT.0) NMQ=NQM(NS1)
79:      IF(NBB.GT.0) NBQ=NQB(NS1)
80:      NXM=6
81:      NXB=10
82:      MA=0
83:      MB=0
84:      IF(NBM.GT.0) MA=1
85:      IF(NBB.GT.0) MB=1
86:      DO 100 INODE=1,NSIDES
87:      DO 100 J=1,5
88:      MQ=0
89:      IF(MAPM(INODE,J).EQ.0) GOTO 500
90:      MQ=MA*MAPM(INODE,J)
91:  500 IF(MAPB(INODE,J).EQ.0) GOTO 400
92:      MQ=MQ+MB*(MAPB(INODE,J)+NMQ)
93:  400 MAPQ(INODE,J)=MQ
94:  100 CONTINUE
95:      IF(NBB.GT.0) GOTO 200
96:      IF(IX34.EQ.0.OR.NSIDES.EQ.3) GOTO 200
97:      NMQ=6
98:      MAPQ(4,3)=6
99:  200 NQT=NMQ+NBQ
100:C
101:      IH=1
102:      NH=NBT*NBT
```

```
      KW=KW-NH
      IT=NH+1
C
      IF(JUMP(1).NE.0) GOTO 150
      CALL HGEN(NBM,NBB,W(IH),W(IT),KW,X,A,XYM,IXYM,XYB,IXYB,IBM,BM
     *,    IBB,BB,NXM,NXB,ISHELL,NBT,IERR)
  150 CONTINUE
C
      NT=NBT*NQT
      IF(NT.GT.KW) GOTO 1000
      NI(1)=1
      NJ(1)=NSIDES
      DO 300 I=2,NSIDES
      NI(I)=NJ(I-1)
  300 NJ(I)=NI(I)-1
C
      IF(JUMP(2).NE.0) GOTO 250
      CALL TGEN(IBM,BM,NBM,IBB,BB,NBB,NSIDES,MAPQ,W(IT),NBT,NQT,X,NI,NJ)
  250 CONTINUE
C
      IF(IX34.GT.0) CALL WARPT(W(IT),MAPQ,NBM,0,NBT,NQT,X,X34)
C
      IF(JUMP(4).NE.0) GOTO 350
      CALL SSTM(W(IH),W(IT),S,STM,NBT,NBT,NQT)
  350 CONTINUE
C
      RETURN
 1000 NEED=NT-KW
      WRITE(6,2000) NEED
 2000 FORMAT( 51H0*** ADDITIONAL CORE REQUIRED FOR ELEMENT K, NEED =
      AERR=4HCORE
      NERR=NEED
      RETURN
      END
```

HGEN

```
      SUBROUTINE HGEN(NBM,NBB,H,W,KW,X,AA,XYM,IXYM,XYB,IXYB,IBM,BM,IBB
     *,    BB,NXM,NXB,ISHELL,NBT,IERR)
      DIMENSION X(2,4),W(1),A(3,3,3),H(NBT,NBT),AA(1)
      COMMON/ERRK2D/AERR,NERR
      COMMON/CK2D4/IQT(7,7),QT(7,7)
      COMMON/CK2D1/XX(4),YY(4),NXD,NYD,DUM
      COMMON/CK2D10/JUMP(10)
      DATA NC,NSM,NSB / 3,3,5 /
      DO 160 I=1,49
  160 IQT(I,1)=0
      DO 260 I=1,4
      XX(I)=X(1,I)
  260 YY(I)=X(2,I)
C
```

```
15:       DO 800 I=1,NBT
16:       DO 800 J=1,NBT
17:   800 H(I,J)=0.0
18:C
19:       NBOTH=0
20:       IF(NBM.GT.0.AND.NBB.GT.0) NBOTH=1
21:       DO 600 I=1,27
22:   600 A(I,1,1)=0.0
23:       IF(NBM.LE.0) GOTO 400
24:       IF(NBM.GT.1) GOTO 500
25:       A(3,3,1)=1.0/AA(1)
26:       GOTO 400
27:   500 K=0
28:       DO 700 J=1,3
29:       DO 700 I=1,J
30:       K=K+1
31:       A(I,J,1)=AA(K)
32:   700 A(J,I,1)=AA(K)
33:   400 IF(NBB.LE.0) GOTO 350
34:       K=6
35:       IF(NBM.LE.0) K=0
36:       DO 900 J=1,3
37:       IF(NBM.GT.0) K=K+3
38:       DO 900 I=1,J
39:       K=K+1
40:       A(I,J,3)=AA(K)
41:   900 A(J,I,3)=AA(K)
42:   350 IF(NBOTH.LE.0) GOTO 150
43:       K=6
44:       DO 250 J=1,3
45:       K=K+J-1
46:       DO 250 I=1,3
47:       K=K+1
48:   250 A(I,J,2)=AA(K)
49:   150 CONTINUE
50:C
51:       NEXT=1
52:       IF(NBM.LE.0) GOTO 100
53:       IPXM=NEXT
54:       IPYM=IPXM+NC*NBM
55:       IPM=IPYM+NC*NBM
56:       NEXT=IPM+NC*NBM
57:       IF(NEXT.GT.KW) GOTO 1000
58:       CALL HGEND(NBM,0,NBT,NC,A(1,1,1),XYM,IXYM,W(IPXM),W(IPYM)
59:      *,    W(IPM),H,NXM,IBM,BM,NSM,0)
60:   100 IF(NBB.LE.0) GOTO 300
61:       IPXB=NEXT
62:       IPYB=IPXB+NC*NBB
63:       IPB=IPYB+NC*NBB
64:       NEXT=IPB+NC*NBB
65:       IF(NEXT.GT.KW) GOTO 1000
66:       CALL HGEND(NBB,NBM,NBT,NC,A(1,1,3),XYB,IXYB,W(IPXB),W(IPYB)
67:      *,    W(IPB),H,NXB,IBB,BB,NSB,0)
68:       IF(ISHELL.EQ.0) GOTO 300
69:       CALL HMBGEN(NBM,NBB,NC,NC,A(1,1,2),W(IPXM),W(IPYM),W(IPM)
```

```
70:      *,    W(IPXB),W(IPYB),W(IPB),H,NBT)
71:  300 CONTINUE
72:      IF(JUMP(3).NE.0) RETURN
73:      CALL SYMVRT(H,NBT,ISING,NBT)
74:      IF(ISING.EQ.0) RETURN
75:      WRITE(6,2000) ISING
76: 2000 FORMAT( 43H0*** WARNING, H MATRIX IS SINGULAR, ISING =  I4 )
77:      IERR=2
78:      RETURN
79: 1000 WRITE(6,3000) KW,NEXT
80: 3000 FORMAT( 38H0*** INSUFFICIENT CORE FOR COMPUTING H  /
81:     *   I10,  9H PROVIDED, I10, 9H REQUIRED )
82:      IERR=1
83:      AERR=4HCORE
84:      NERR=NEXT
85:      RETURN
86:      END
```

HGEND

```
 1:      SUBROUTINE HGEND(NB,INCB,NBT,NC,C,XY,IXY,IPX,IPY,P,H,NX,IB,B,NS
 2:     *,    INCC)
 3:      COMMON/CK2D4/IQT(7,7),QT(7,7)
 4:      COMMON/CK2D1/XX(4),YY(4),MX,MY,Z
 5:      DIMENSION C(NC,NC),IPX(NC,NB),IPY(NC,NB),P(NC,NB)
 6:     *,    H(NBT,NBT),XY(NX),IXY(2,NX)
 7:     *,    IB(NS,NX),B(NS,NX)
 8:C
 9:      ZERO=1.0E-10
10:C
11:      DO 400 I=1,NC
12:      DO 400 J=1,NB
13:      IPX(I,J)=0
14:      IPY(I,J)=0
15:  400 P(I,J)=0.0
16:      DO 500 IC=1,NC
17:      IIC=IC+INCC
18:      DO 500 IX=1,NX
19:      IIB=IB(IIC,IX)
20:      IF(IIB.LE.0) GOTO 500
21:      IF(IIB.GT.NB) GOTO 500
22:      BIJ=B(IIC,IX)
23:      P(IC,IIB)=BIJ*XY(IX)
24:      IPX(IC,IIB)=IXY(1,IX)
25:      IPY(IC,IIB)=IXY(2,IX)
26:  500 CONTINUE
27:C
28:      DO 100 I=1,NB
29:      IH=I+INCB
30:      DO 100 J=1,I
```

```
31:       JH=J+INCB
32:       DO 200 IC=1,NC
33:       DO 200 JC=1,NC
34:       PC=P(IC,I)*C(IC,JC)*P(JC,J)
35:       IF(ABS(PC).LT.ZERO) GOTO 200
36:       MX=IPX(IC,I)+IPX(JC,J)
37:       MY=IPY(IC,I)+IPY(JC,J)
38:       MMX=MX+1
39:       MMY=MY+1
40:       IF(IQT(MMX,MMY).EQ.1) GOTO 600
41:       CALL ITQUAD
42:       IQT(MMX,MMY)=1
43:       QT(MMX,MMY)=Z
44:   600 H(IH,JH)=H(IH,JH)+PC*QT(MMX,MMY)
45:   200 CONTINUE
46:       H(JH,IH)=H(IH,JH)
47:   100 CONTINUE
48:C
49:       RETURN
50:       END
```

HMBGEN

```
 1:       SUBROUTINE HMBGEN(NBM,NBB,NCM,NCB,C,IPXM,IPYM,PM,IPXB,IPYB,PB
 2:      *,    H,NBT )
 3:       DIMENSION C(NCM,NCB),IPXM(NCM,NBM),IPYM(NCM,NBM)
 4:      *,    PM(NCM,NBM),IPXB(NCB,NBB),IPYB(NCB,NBB),PB(NCB,NBB)
 5:      *,    H(NBT,NBT)
 6:       COMMON/CK2D4/IQT(7,7),QT(7,7)
 7:       COMMON/CK2D1/XX(4),YY(4),NX,MY,Z
 8:C
 9:       ZERO=1.0E-10
10:C
11:C
12:       DO 100 I=1,NBM
13:       IH=I
14:       DO 100 J=1,NBB
15:       JH=J+NBM
16:       DO 200 IC=1,NCM
17:       DO 200 JC=1,NCB
18:       PC=PM(IC,I)*C(IC,JC)*PB(JC,J)
19:       IF(ABS(PC).LT.ZERO) GOTO 200
20:       NX=IPXM(IC,I)+IPXB(JC,J)
21:       MY=IPYM(IC,I)+IPYB(JC,J)
22:       NNX=NX+1
23:       MMY=MY+1
24:       IF(IQT(NNX,MMY).EQ.1) GOTO 600
25:       CALL ITQUAD
```

```
26:       IQT(NNX,MMY)=1
27:       QT(NNX,MMY)=Z
28:   600 H(IH,JH)=H(IH,JH)+PC*QT(NNX,MMY)
29:   200 CONTINUE
30:   100 H(JH,IH)=H(IH,JH)
31:       RETURN
32:       END
```

SYMVRT

```
 1:       SUBROUTINE SYMVRT(A,N,ISING,NHD)
 2:C      SYMETRIC INVERSION SUBROUTINE
 3:       DIMENSION A(NHD,NHD),B(30),T(30),ERR(2)
 4:       COMMON/ERRK2D/AERR,NERR
 5:       DATA ERR(1),ERR(2)/4HISNG,4HNEGD/
 6:       IF(N.GT.1) GOTO 100
 7:       A(1,1)=1.0/A(1,1)
 8:       RETURN
 9:   100 CONTINUE
10:       IX=N
11:       ISING=0
12:       DO 18 ID=1,IX
13:       T(ID)=1.
14:       IDM1=ID-1
15:       IDP1=ID+1
16:       IF(IDM1)10,14,10
17:    10 DO 11 J=ID,IX
18:       DO 11 K=1,IDM1
19:    11 A(ID,J)=A(ID,J)-A(K,ID)*A(K,J)*T(K)
20:    14 IF(A(ID,ID))13,12,15
21:    12 ISING=ID
22:       NERR=ID
23:       AERR=ERR(1)
24:       GO TO 30
25:    13 T(ID)=-1.
26:       NERR=NERR+1
27:       AERR=ERR(2)
28:       A(ID,ID) = ABS(A(ID,ID))
29:    15 A(ID,ID)=SQRT (A(ID,ID))*T(ID)
30:       IF(IDP1-IX)16,16,19
31:    16 DO 17 J=IDP1,IX
32:    17 A(ID,J)=A(ID,J)/A(ID,ID)
33:       A(ID,ID)=A(ID,ID)*T(ID)
34:    18 CONTINUE
35:    19 A(IX,IX)=A(IX,IX)*T(IX)
36:       IXM1=IX-1
37:       DO 24 I=1,IXM1
38:       I1=I+1
```

```
39:       DO 20 J=I1,IX
40:    20 A(I,J)=A(I,J)/A(I,I)
41:    24 A(I,I)=T(I)/A(I,I)
42:       A(IX,IX)=T(IX)/A(IX,IX)
43:       DO 23 I=2,IX
44:       K=I+1
45:       IM1=I-1
46:       DO 23 L=1,IM1
47:       IF(K-IX)21,21,23
48:    21 DO 22 J=K,IX
49:    22 A(L,J)=A(L,J)-A(I,J)*A(L,I)
50:    23 A(L,I)=-A(L,I)*A(I,I)
51:       DO 27 I=1,IX
52:       B(I)=0.
53:       DO 29 K=I,IX
54:    29 B(I)=B(I)+A(I,K)**2*T(K)
55:       M=I+1
56:       IF(M-IX)26,26,32
57:    26 DO 27 J=M,IX
58:       A(J,I)=0.
59:       DO 27 K=J,IX
60:    27 A(J,I)=A(J,I)+A(I,K)*A(J,K)*T(K)
61:    32 DO 28 I=1,IX
62:    28 A(I,I)=B(I)
63:       DO 35  J=1,N
64:       DO 35  I=1,N
65: 35    A(J,I)=A(I,J)
66:    30 RETURN
67:       END
```

ITQUAD

```
 1:       SUBROUTINE ITQUAD
 2:       COMMON/CK2D1/ X(4),Y(4),N,M,AXNYM
 3:       COMMON/CK2D2/ND,MD,MMD,CC(50),C(50),T,SUM
 4:       COMMON/CK2D3/AR(3)
 5:       COMMON/ZROK2D/ZERO
 6:       ND=N
 7:       DO 100 I=1,3
 8:  100 AR(I)=0.0
 9:C* INTEGRATE ZONE 1
10:       X3MX4=X(3)-X(4)
11:       XL=X3MX4
12:       IF(X(2).GT.XL) XL=X(2)
13:       XM=1.0/XL
14:       IF(ABS(X3MX4*XM).LT.ZERO) GOTO 200
15:       A=Y(4)
16:       B=(Y(3)-Y(4))/X3MX4
17:       AM=1.0
18:       MD=M+1
```

```
      IF(ABS(B).GT.ZERO) GOTO 300
      MD=0
      C(1)=1.0
      AM=A**(M+1)
      GOTO 700
C* EXPAND THE Y BINOMIAL
  300 AMP2=MD+2
      ADB=A/B
      C(1)=B**MD
      M1=MD+1
      DO 150 I=2,M1
      AI1=I
      AI=(AMP2-AI1)/(AI1-1.0)
      C(I)=C(I-1)*AI*ADB
  150 CONTINUE
C* EXPAND THE X BINOMIAL
  700 AMP2=N+2
      ADB=X(4)
      CC(1)=1.0
      IF(N.EQ.0) GOTO 450
      N1=N+1
      DO 250 I=2,N1
      AI1=I
      AI=(AMP2-AI1)/(AI1-1.0)
      CC(I)=CC(I-1)*AI*ADB
  250 CONTINUE
  450 MMD=MD+1
      T=X3MX4
      CALL DSUM
      AR(1)=SUM
      AR(1)=AR(1)*AM
C* INTEGRATE ZONE 2
  200 IF(ABS(X(4)*XM).LT.ZERO) GOTO 400
      ANM=N+M+2
      AR(2)=(X(4)**(N+M+2))*(Y(4)/X(4))**(M+1)
      AR(2)=AR(2)/ANM
C* INTEGRATE ZONE 3
  400 X3MX2=X(3)-X(2)
      IF(ABS(X3MX2*XM).LT.ZERO) GOTO 600
C* EXPAND THE X BINOMIAL
      AMP2=N+2
      ADB=X(2)
      CC(1)=1.0
      IF(N.EQ.0) GOTO 550
      N1=N+1
      DO 350 I=2,N1
      AI1=I
      AI=(AMP2-AI1)/(AI1-1.0)
      CC(I)=CC(I-1)*AI*ADB
  350 CONTINUE
  550 C(1)=1.0
      MD=M+1
      MMD=1
      T=X3MX2
      CALL DSUM
```

```
74:C* SUM 3 ZONES
75:       AR(3)=-SUM*(Y(3)/X3MX2)**(M+1)
76:   600 CONTINUE
77:       AXNYM=0
78:       DO 800 I=1,3
79:   800 AXNYM=AXNYM+AR(I)
80:       AM1=M+1
81:       AXNYM=AXNYM/AM1
82:       RETURN
83:       END
```

DSUM

```
 1:       SUBROUTINE DSUM
 2:       COMMON/CK2D2/N,M,MM,B(50),C(50),T,SUM
 3:       SUM=0.
 4:       N1=N+1
 5:       M1=MM
 6:       IB=1
 7:       NE=1
 8:       NM=N+M+1
 9:       IE=NM
10:       DO 100 ITERM=1,NM
11:       NE=NE+1
12:       J1=NE-IB
13:       IF(J1.GT.M1) IB=IB+1
14:       IF(IB.GT.N1) GOTO 100
15:       IE=IE-1
16:       D=0.
17:       DO 200 I=IB,N1
18:       J=NE-I
19:       IF(J.EQ.0) GOTO 300
20:   200 D=D+B(I)*C(J)
21:   300 AE1=IE+1
22:       IEE=IE+1
23:       SUM=SUM+(D*T**IEE)/AE1
24:   100 CONTINUE
25:       RETURN
26:       END
```

```
      SUBROUTINE TGEN(IBM,BM,NBM,IBB,BB,NBB,NSIDES,MAPQ
     *,    T,NBT,NQT,X,NI,NJ)
      DIMENSION IBM(3,1),BM(3,1),IBB(5,1),BB(5,1)
     *,    T(NBT,NQT),X(2,1),NI(1),NJ(1)
     *,    NMA(2,9),NMC(2,9),TT(100),XNCNA(9,9)
     *,    R(5),A(9,8),IDQ(4),MAPU(3,5),NIJ(2)
     *,    C(150),TEMP(250)
C
      COMMON/WARP/ IWARP,NALFA(4),ALFA(5,4),NGAMA(4),GAMA(5,4)
      COMMON/CK2D10/JUMP(10)
      COMMON/CK2D5/IQX(10),QX(10)
C
      DATA ((MAPU(I,J),I=1,3),J=1,5) /  3, 0,-2, 1, 2, 0, 6, 0,-5,
     *                                  4, 5, 0, 2,-1, 3 /
C
      TH=1.0
C
      DO 100 I=1,NBT
      DO 100 J=1,NQT
  100 T(I,J)=0.0
      DO 200 ISIDE=1,NSIDES
      DO 160 I=1,10
  160 IQX(I)=0
      II=NI(ISIDE)
      JJ=NJ(ISIDE)
      T1=X(1,II)
      T3=X(2,II)
      ZL=0.0
      DO 300 I=1,2
  300 ZL=ZL+(X(I,JJ)-X(I,II))**2
      ZL=SQRT(ZL)
      CS=(X(1,JJ)-X(1,II))/ZL
      SN=(X(2,JJ)-X(2,II))/ZL
C
      NODES=2
      NIJ(1)=II
      NIJ(2)=JJ
C
      DO 500 IP=1,5
      IF(JUMP(5).NE.0) GOTO 700
      IF(IP.GT.2) GOTO 600
      IF(NBM.EQ.0) GOTO 500
      NBC=NBM
      IF(IP.EQ.1) CALL TTGEN(TT,T1,CS,T3,SN,NX,NC,NMC,1)
      INCB=0
      CALL RGEN(R,IP,SN,CS)
      CALL TCB(3,NX,R,TT,BM,IBM,C,NC,NBM)
      GOTO 700
  600 IF(NBB.EQ.0) GOTO 500
      NBC=NBB
```

```
51:       IF(IP.EQ.3) CALL TTGEN(TT,T1,CS,T3,SN,NX,NC,NMC,2)
52:       INCB=NBM
53:       CALL RGEN(R,IP,SN,CS)
54:       CALL TCB(5,NX,R,TT,BB,IBB,C,NC,NBB)
55:   700 CONTINUE
56:       DO 800 IUU=1,3
57:       IU=MAPU(IUU,IP)
58:       IF(IUU.EQ.1) GOTO 900
59:       IF(IWARP.EQ.0) GOTO 800
60:       IF(IUU.EQ.2.AND.NALFA(ISIDE).EQ.0) GOTO 800
61:       IF(IUU.EQ.3.AND.NGAMA(ISIDE).EQ.0) GOTO 800
62:       IF(IU.EQ.0) GOTO 800
63:       IF(IU.GT.0) SIGN=+1.0
64:       IF(IU.LT.0) SIGN=-1.0
65:       IU=IABS(IU)
66:       IF(IP.LT.3.AND.NBB.EQ.0) IU=7
67:   900 IF(JUMP(6).NE.0) GOTO 150
68:       CALL AGEN(A,NMA,NA,NQA,NQ1,IDQ,ZL,CS,SN,IU)
69:   150 IF(JUMP(7).NE.0) GOTO 250
70:       IF(IUU.EQ.1) CALL NCALNA(ZL,XNCNA,NMC,NMA,NC,NA,
71:      *    0,ALFA,SIGN)
72:       IF(IUU.EQ.2) CALL NCALNA(ZL,XNCNA,NMC,NMA,NC,NA,
73:      *    NALFA(ISIDE),ALFA(1,ISIDE),SIGN)
74:       IF(IUU.EQ.3) CALL NCALNA(ZL,XNCNA,NMC,NMA,NC,NA,
75:      *    NGAMA(ISIDE),GAMA(1,ISIDE),SIGN)
76:   250 IF(JUMP(8).NE.0) GOTO 800
77:       CALL CXA(C,XNCNA,A,T,TEMP,NA,NQA,NQT,NC,NBC,NBT,INCB,MAPQ
78:      *,   IDQ,NODES,NIJ,NQ1)
79:   800 CONTINUE
80:   500 CONTINUE
81:   200 CONTINUE
82:       RETURN
83:       END
```

TTGEN

```
 1:       SUBROUTINE TTGEN (TT,T1,T2,T3,T4,NX,NC,NMC,IKIND)
 2:       DIMENSION TT(1),NMC(2,1)
 3:       GOTO(10,20),IKIND
 4:    10 NX=6
 5:       NC=3
 6:       CALL TT6X3(T1,T2,T3,T4,TT)
 7:       GOTO 100
 8:    20 NX=10
 9:       NC=3
10:       CALL TT10X3(T1,T2,T3,T4,TT)
11:   100 CONTINUE
12:       NXP=-1
13:       DO 30 I=1,NC
14:       NXP=NXP+1
```

```
15:       NMC(1,I)=NXP
16:    30 NMC(2,I)=0
17:       RETURN
18:       END
```

TT6X3

```
 1:       SUBROUTINE TT6X3(T1,T2,T3,T4,T)
 2:       DIMENSION  T(6,3)
 3:       DO 150 I=1,6
 4:       DO 150 J=1,3
 5:   150 T(I,J)=0.0
 6:       T(1,1)=1.0
 7:       T(2,1)=T1
 8:       T(3,1)=T3
 9:       T(4,1)=T1**2
10:       T(5,1)=T3**2
11:       T(6,1)=T1*T3
12:       T(2,2)=T2
13:       T(3,2)=T4
14:       T(4,2)=2.0*T1*T2
15:       T(5,2)=2.0*T3*T4
16:       T(6,2)=T1*T4+T2*T3
17:       T(4,3)=T2**2
18:       T(5,3)=T4**2
19:       T(6,3)=T2*T4
20:       RETURN
21:       END
```

TT10X3

```
 1:       SUBROUTINE TT10X3(T1,T2,T3,T4,TT)
 2:       DIMENSION TT(10,3)
 3:       DO 200 I=1,10
 4:       DO 200 J=1,3
 5:   200 TT(I,J)=0.0
 6:       TT(1,1)=1.0
 7:       TT(3,1)=T1
 8:       TT(3,2)=T2
 9:       TT(5,1)=T3
10:       TT(5,2)=T4
11:       TT(7,1)=T1**2
12:       TT(7,2)=2.*T1*T2
13:       TT(7,3)=T2**2
14:       TT(8,1)=T3**2
```

```
      TT(8,2)=2.*T3*T4
      TT(8,3)=T4**2
      TT(9,1)=T1*T3
      TT(9,2)=T1*T4+T2*T3
      TT(9,3)=T2*T4
      DO 300 I=1,4
      II=2*I
      IF(I.EQ.4) II=10
      DO 300 J=1,3
  300 TT(II,J)=TT(II-1,J)
      RETURN
      END
```

RGEN

```
      SUBROUTINE RGEN(R,IP,SN,CS)
      DIMENSION R(1)
      GOTO(10,20,30,40,50),IP
   10 R(1)=-SN**2
      R(2)=-CS**2
      R(3)=2.0*SN*CS
      GOTO 100
   20 R(1)=-SN*CS
      R(2)=-R(1)
      R(3)=CS**2-SN**2
      GOTO 100
   30 R(1)=SN*CS
      R(2)=-SN*CS
      R(3)=CS**2-SN**2
      R(4)=0.0
      R(5)=0.0
      GOTO 100
   40 R(1)=-SN**2
      R(2)=-CS**2
      R(3)=-2.*SN*CS
      R(4)=0.0
      R(5)=0.0
      GOTO 100
   50 DO 200 I=1,3
  200 R(I)=0.0
      R(4)=-SN
      R(5)=CS
  100 CONTINUE
      RETURN
      END
```

TCB

```
 1:       SUBROUTINE TCB(NI,NJ,R,Z,B,IB,XCB,NC,NB)
 2:       DIMENSION R(1),Z(NJ,NC),B(NI,NJ),IB(NI,NJ),XCB(NC,NB)
 3:C
 4:       DO 100 IC=1,NC
 5:       DO 200 I=1,NB
 6:   200 XCB(IC,I)=0.0
 7:       DO 300 I=1,NI
 8:       DO 300 J=1,NJ
 9:       IIB=IB(I,J)
10:       IF(IIB.EQ.0) GOTO 300
11:       XCB(IC,IIB)=XCB(IC,IIB)+R(I)*Z(J,IC)*B(I,J)
12:   300 CONTINUE
13:   100 CONTINUE
14:       RETURN
15:       END
```

AGEN

```
 1:       SUBROUTINE AGEN(A,NMA,NA,NQA,NQ1,IDQ,ZL,CS,SN,IU)
 2:       DIMENSION A(1),NMA(2,1),IDQ(1),AB(2,4,2),D(2,4)
 3:C
 4:       DO 100 I=1,2
 5:       DO 100 J=1,4
 6:       D(I,J)=0.0
 7:       DO 100 K=1,2
 8:   100 AB(I,J,K)=0.0
 9:       GOTO(10,20,30,40,50,60,70),IU
10:C
11:    10 NA=2
12:       NQ1=2
13:       NQ1B=1
14:       IDQ(1)=1
15:       IDQ(2)=2
16:       AB(1,1,1)=1.0
17:       AB(1,2,1)=-1.0/ZL
18:       AB(2,2,1)=1.0/ZL
19:       D(1,1)=CS
20:       D(1,2)=SN
21:       GOTO 200
22:    20 NA=4
23:       NQ1=3
24:       NQ1B=2
25:       IDQ(1)=3
26:       IDQ(2)=4
27:       IDQ(3)=5
```

```
28:       ZL2=ZL*ZL
29:       ZL3=ZL2*ZL
30:       AB(1,1,1)=1.0
31:       AB(1,3,1)=-3.0/ZL2
32:       AB(1,4,1)=2.0/ZL3
33:       AB(1,2,2)=1.0
34:       AB(1,3,2)=-2.0/ZL
35:       AB(1,4,2)=1.0/ZL2
36:       AB(2,3,1)=-AB(1,3,1)
37:       AB(2,4,1)=-AB(1,4,1)
38:       AB(2,3,2)=-1.0/ZL
39:       AB(2,4,2)=1.0/ZL2
40:       D(1,1)=1.0
41:       D(2,2)=SN
42:       D(2,3)=-CS
43:       GOTO 200
44:    30 NA=2
45:       NQ1=2
46:       NQ1B=1
47:       IDQ(1)=1
48:       IDQ(2)=2
49:       AB(1,1,1)=1.0
50:       AB(1,2,1)=-1.0/ZL
51:       AB(2,2,1)=1.0/ZL
52:       D(1,1)=SN
53:       D(1,2)=-CS
54:       GOTO 200
55:    40 NA=2
56:       NQ1=2
57:       NQ1B=1
58:       IDQ(1)=4
59:       IDQ(2)=5
60:       AB(1,1,1)=1.0
61:       AB(1,2,1)=-1.0/ZL
62:       AB(2,2,1)=1.0/ZL
63:       D(1,1)=CS
64:       D(1,2)=SN
65:       GOTO 200
66:    50 NA=1
67:       NQ1=2
68:       NQ1B=1
69:       IDQ(1)=1
70:       IDQ(2)=2
71:       AB(1,1,1)=1.0/ZL
72:       AB(2,1,1)=-1.0/ZL
73:       D(1,1)=SN
74:       D(1,2)=-CS
75:       GOTO 200
76:    60 NA=3
77:       NQ1=3
78:       NQ1B=2
79:       IDQ(1)=3
80:       IDQ(2)=4
81:       IDQ(3)=5
82:       ZL2=ZL*ZL
```

```
      ZL3=ZL2*ZL
      AB(1,2,1)=-6.0/ZL2
      AB(1,3,1)=6.0/ZL3
      AB(1,1,2)=1.0
      AB(1,2,2)=-4.0/ZL
      AB(1,3,2)=3.0/ZL2
      AB(2,2,1)=6.0/ZL2
      AB(2,3,1)=-6.0/ZL3
      AB(2,2,2)=-2.0/ZL
      AB(2,3,2)=3.0/ZL2
      D(1,1)=1.0
      D(2,2)=SN
      D(2,3)=-CS
      GOTO 200
   70 NA=2
      NQ1=1
      NQ1B=1
      IDQ(1)=3
      AB(1,1,1)=1.0
      AB(1,2,1)=-1.0/ZL
      AB(2,2,1)=1.0/ZL
      D(1,1)=1.0
C
  200 CONTINUE
      NM=-1
      DO 300 I=1,NA
      NM=NM+1
      NMA(1,I)=NM
  300 NMA(2,I)=0
      NQA=2*NQ1
      CALL ATD(A,AB,D,NA,NQA,NQ1B,NQ1)
      RETURN
      END
```

ATD

```
      SUBROUTINE ATD(A,AB,D,NA,NQA,NQ1B,NQ1)
      DIMENSION A(NA,NQA), AB(2,4,NQ1B),D(2,4)
      JST=-NQ1
      DO 100 IEND=1,2
      JST=JST+NQ1
      DO 200 IA=1,NA
      DO 200 J=1,NQ1
      JJ=JST+J
      A(IA,JJ)=0.0
      DO 200 K=1,NQ1B
      A(IA,JJ)=A(IA,JJ)+AB(IEND,IA,K)*D(K,J)
  200 CONTINUE
  100 CONTINUE
      RETURN
      END
```

NCALNA

```
      SUBROUTINE NCALNA(XL,Z,NMC,NMA,NC,NA,NALFA,ALFA,SIGN)
      DIMENSION Z(NC,NA),NMC(2,NC),NMA(2,NA),ALFA(1)
      COMMON/CK2D5/IQX(10),QX(10)
      DO 100 I=1,NC
      DO 100 J=1,NA
      N=NMC(1,I)+NMA(1,J)
      IF(NALFA.GT.0) GOTO 200
      N1=N+1
      IF(IQX(N1).GT.0) GOTO 400
      IQX(N1)=1
      AN1=N1
      QX(N1)=(XL**N1)/AN1
400   Z(I,J)=QX(N1)
      GOTO 100
200   Z(I,J)=0.0
      N=N-1
      DO 300 IAL=1,NALFA
      N=N+1
      N1=N+1
      IF(IQX(N1).GT.0) GOTO 300
      IQX(N1)=1
      AN1=N1
      QX(N1)=(XL**N1)/AN1
300   Z(I,J)=Z(I,J)+QX(N1)*ALFA(IAL)*SIGN
100   CONTINUE
      RETURN
      END
```

CXA

```
      SUBROUTINE CXA(C,X,A,T,TMP,NA,NQA,NQT,NC,NBC,NBT,INCB,MAPQ,IDQ
     *,   NODES,NIJ,NQ1)
      DIMENSION          X(NC,NA),A(NA,NQA),T(NBT,NQT),TMP(NBC,NA)
     *,  C(NC,NBC),IDQ(NQ1),NIJ(NODES),MAPQ(4,1)
C
C***  THIS ROUTINE COMPUTES T=C(TRANSPOSE)*X*A
C***  RESULT IS ACCUMULATED.  T MUST BE INITIALIZED OUTSIDE
C
      DO 100 I=1,NBC
      DO 100 J=1,NA
```

```
11:       TMP(I,J)=0.0
12:       DO 100 K=1,NC
13:   100 TMP(I,J)=TMP(I,J)+C(K,I)*X(K,J)
14:C
15:       DO 200 I=1,NBC
16:       J=0
17:       IT=I+INCB
18:       DO 200 ND=1,NODES
19:       NODE=NIJ(ND)
20:       DO 200 IQ=1,NQ1
21:       IIQ=IDQ(IQ)
22:       J=J+1
23:       JT=MAPQ(NODE,IIQ)
24:       IF(JT.LE.0) GO TO 200
25:       DO 300 K=1,NA
26:   300 T(IT,JT)=T(IT,JT)+TMP(I,K)*A(K,J)
27:   200 CONTINUE
28:       RETURN
29:       END
```

. SSTM

```
 1:       SUBROUTINE SSTM(H,T,S,STM,NHD,NB,NQ)
 2:       DIMENSION H(NHD,NHD),T(NHD,NQ),S(1),STM(NB,NQ)
 3:       DO 100 I=1,NB
 4:       DO 100 J=1,NQ
 5:       STM(I,J)=0.0
 6:       DO 100 K=1,NB
 7:   100 STM(I,J)=STM(I,J)+H(I,K)*T(K,J)
 8:       K=0
 9:       DO 200 I=1,NQ
10:       DO 200 J=1,I
11:       K=K+1
12:       S(K)=0.0
13:       DO 200 L=1,NB
14:   200 S(K)=S(K)+T(L,I)*STM(L,J)
15:       RETURN
16:       END
```

Experimental Element Capability

Table of Contents

EXPE 1.0 - TERMINOLOGY:

FOR EACH ELEMENT, AN "ELEMENT REFERENCE FRAME" IS DEFINED USING THE SAME CONVENTION AS FOR STANDARD SPAR ELEMENT TYPES E31, E41, S41, S61, AND S81; THAT IS, THE X AXIS IS DIRECTED FROM NODE 1 THROUGH NODE 2, AND NODE 3 LIES IN THE FIRST QUADRANT OF THE X-Y PLANE.

THE FOLLOWING SYMBOLS WILL BE USED IN EXPLAINING HOW TO EMPLOY THE EXPERIMENTAL ELEMENT CAPABILITY:

NAME	DIMENSION	DEFINITION
TYPE		A 4-CHARACTER ALPHANUMERIC WORD ASSIGNED BY THE USER TO IDENTIFY A PARTICULAR ELEMENT FORMULATION, ANALOGOUS TO THE STANDARD ELEMENT TYPE CODES (E21, E43, S81, ETC.). A MODEL MAY CONSIST OF ANY OF THE STANDARD ELEMENT TYPES, PLUS ONE OR MORE TYPES OF EXPERIMENTAL ELEMENTS. THERE IS NO SPECIFIC LIMIT ON THE NUMBER OF TYPES OF EXPERIMENTAL ELEMENTS.
MAJOR		ELEMENTS ARE CLASSIFIED AS: MAJOR = 1 FOR LINE ELEMENTS, E.G. E21, E22. MAJOR = 2 FOR 2D ELEMENTS, E.G. E31, E43. MAJOR = 3 FOR 3D ELEMENTS, E.G. S81, F41.
MINOR		AN INTEGER ASSIGNED BY THE USER TO IDENTIFY A SPECIFIC ELEMENT TYPE. A UNIQUE "MINOR" IS DEFINED FOR EACH "TYPE."
N		THE NUMBER OF NODES PER ELEMENT (MINIMUM = 3, MAXIMUM = 32). EXPERIMENTAL BEAM ELEMENTS SHOULD BE MODELLED AS 3-NODE ELEMENTS, USING THIRD POINT TO ESTABLISH THE CROSS-SECTION ORIENTATION.
M		N(N + 1)/2. SEE K, CM, KG BELOW.
NDF		NUMBER OF DEGREES OF FREEDOM PER NODE, EITHER 3 (3 DISPLACEMENTS), OR 6 (3 DISPLACEMENTS AND 3 ROTATIONS).
NNDF		THE TOTAL NUMBER OF DEGREES OF FREEDOM PER ELEMENT, N TIMES NDF. SEE U AND F BELOW.
X	(3,N)	RELATIVE TO THE ELEMENT REFERENCE FRAME, THE DIRECTION I POSITION COORDINATE OF ELEMENT NODE J IS X(I,J).

TERMINOLOGY, CONTINUED:

NAME	DIMENSION	DEFINITION
U	(NNDF)	ELEMENT NODAL MOTION VECTOR. WHERE DIJ AND RIJ ARE DIRECTION I DISPLACEMENT AND ROTATION COMPONENTS (RELATIVE TO THE ELEMENT REFERENCE FRAME) OF ELEMENT NODE J, THE ORDER OF TERMS IN U IS AS FOLLOWS, IF NDF=6: D11 D21 D31 R11 R21 R31 D12 D22 D32 R12 - - IF NDF = 3, THE ORDER IS THE SAME, EXCEPT THE ROTATION TERMS ARE NOT PRESENT. NONE OF THE USER-WRITTEN SUBROUTINES ACCESS EITHER U OR F, DEFINED BELOW. U AND F ARE DEFINED HERE ONLY FOR USE IN DEFINING OTHER ARRAYS WHICH MUST BE GENERATED IN USER-WRITTEN ROUTINES.
F	(NNDF)	ELEMENT NODAL FORCE VECTOR, CORRESPONDING TO U.
P N4PROP	(NP)	ELEMENT PROPERTY ARRAY. THE CONTENT OF P IS ESTABLISHED BY THE USER. TYPICAL ITEMS ARE MATERIAL AND SECTION PROPERTIES, OPTION CONTROLLERS, ETC. BEFORE EXECUTING ELD, THE USER MUST CREATE, VIA AUS/TABLE, A TABLE NAMED XXXX BTAB 2 N4PROP, WHERE N4PROP IS ANY INTEGER GREATER THAN 100. EACH LINE IN THIS TABLE IS A P ARRAY, APPLICABLE TO ONE OR MORE INDIVIDUAL ELEMENTS, AS ESTABLISHED VIA THE NSECT POINTER IN ELD.
S	(NS)	THE ELEMENT "STRESS" STATE VECTOR. THE CONTENT OF S IS ESTABLISHED BY THE USER. TYPICAL TERMS IN S ARE STRESS FIELD COEFFICIENTS, STRAINS, STRESS RESULTANTS, ETC. SEE THE DEFINITION OF Q, R, AND D BELOW.
Q	(ND)	THE ELEMENT "THERMAL" LOAD VECTOR. THE CONTENT OF Q IS ESTABLISHED BY THE USER. TYPICAL COMPONENTS OF Q ARE COEFFICIENTS OF TEMPERATURE OR DISLOCATION FUNCTIONS, TEMPERATURE GRADIENTS, ETC., DEFINED IN ANY MANNER THE USER FINDS CONVENIENT. SEE THE DEFINITION OF THE C AND D MATRICES BELOW. THE Q VECTORS FOR INDIVIDUAL ELEMENTS ARE CONSTRUCTED VIA AUS/ELDATA. THE NODAL TEMPerature AND NODAL PRESsure ARRAYS , IF PRESENT, DO NOT APPLY TO EXPERIMENTAL ELEMENTS.

TERMINOLOGY, CONTINUED:

NAME	DIMENSION	DEFINITION
Y	(N)	ELEMENT NODAL WEIGHT DISTRIBUTION. THE WEIGHT OF AN ELEMENT = Y(1) + Y(2) + - - - Y(N). PROCESSOR E USES Y IN CONSTRUCTING THE SYSTEM DIAGONAL MASS MATRIX, DEM.
K	(NDF,NDF,M)	THE ELEMENT STIFFNESS MATRIX, ARRAYED AS M SUBMATRICES, EACH DIMENSIONED NDF BY NDF:

$$K = \begin{bmatrix} K(-,-,1) & K(-,-,2) & K(-,-,4) & - & - & - \\ & K(-,-,3) & K(-,-,5) & - & - & - \\ & & K(-,-,6) & - & - & - \\ & & & & - & - \\ \text{SYMMETRIC} & & & & & K(-,-,M) \end{bmatrix}$$

C	(NNDF,ND)	THERMAL FORCE INFLUENCE MATRIX. TOTAL ELEMENT NODAL FORCES ARE: $F = K\,U + C\,Q$
P	(NS,NNDF)	MECHANICAL STRESS RECOVERY MATRIX.
D	(NS,ND)	THERMAL STRESS RECOVERY MATRIX. THE TOTAL STRESS, AS COMPUTED BY GSF, IS: $S = P\,U + D\,Q$
CM	(NDF,NDF,M)	ELEMENT CONSISTENT MASS MATRIX, ARRAYED THE SAME AS K.
KG	(NDF,NDF,M)	ELEMENT INITIAL-STRESS STIFFNESS MATRIX, ARRAYED THE SAME AS K.

EXPE 2.0 - USER WRITTEN SUBROUTINES:

To implement experimental elements, the user must code the following routines and incorporate them into the indicated processors, replacing empty routines having the same names in the standard version of SPAR. It is permissible to omit DMEXPE, CMEXPE, and KGEXPE.

Processor	Routine Name	Argument List	
		Inputs:	Outputs:
E	DMEXPE(	MAJOR,MINOR,N,NDF,P,X,	Y)
EKS	KEXPE(	MAJOR,MINOR,N,NDF,P,X,	K,P,C,D)
M	CMEXPE(	MAJOR,MINOR,N,NDF,P,X,	CM)
KG	KGEXPE(	MAJOR,MINOR,N,NDF,P,X,S,	KG)

The primary limitation on the experimental element capability is the total length of an element information packet in the E-state, which is initiated in processor E. The component parts are:

Part	Length
1	12 + N
2	0
3	9 + 12N
4 (P)	NP
5 (K)	M X NDF X NDF
6 (R)	NS X NNDF
7 (S)	NS
8 (C)	NNDF X ND
9 (D)	NS X ND

The related limitations are:

- The sum of the lengths of parts 1 through 4 may not exceed 1000.
- All processors accessing the E-state (e.g., K, M, KG, EQNF, GSF) require sufficient central memory to load at least one complete element information packet, in addition to the other C.M. requirements, as defined in the SPAR Reference Manual.
- In E, the entire XXXX BTAB 2 N4PROP array must be in central memory.

Although the experimental element capability has been structured such that there are few size restrictions, users should be aware that extremely large element information packets will generally result in relatively high I/O costs.

EXPE 3.0 - PROGRAM EXECUTION:

Executions proceed the same as in a standard analysis, except that before executing ELD, the user must execute AUS/TABLE to construct, for each type of experimental element, a table named XXXX BTAB 2 N4PROP. The identifying integer, N4PROP, may be any number greater than 100. Each line in this table is a P array, as previously defined, containing the material and section constants applicable to one or more elements.

It is also mandatory that a MATERIAL CONSTANT data set be generated in TAB, even though the experimental elements do not reference MATC. This is required because processor E automatically loads MATC, and retains it in central memory throughout execution.

Experimental elements are defined in ELD as illustrated below. Within the same ELD execution, other element types, either standard or experimental, may also be defined.

```
@XQT ELD
$
% Define all elements of a given Type:
$
 EXPE Type, major, minor, N, NDF, NS, NG, N4PROP$
 NSECT=K      $ Line K of XXXX BTAB 2 N4PROP is "P" for
$               elements defined subsequently.
 J1 J2 - - JN$ First element connects joints J1 J2 - -JN.
 J1 J2 - - JN$ Second element
-
-
 NSECT=J      $ Line J of XXXX BTAB 2 N4PROP is "P" for
$               elements defined subsequently.
 J1 J2 - - JN$
 J1 J2 - - JN$
-
-
```

NSECT is the only table pointer command applicable to experimental elements. The MOD, INC, and GROUP commands function the same as for the standard elements. The standard mesh generation facilities may be used for experimental elements having the same nodal arrangement as standard elements.

If thermal loading is present, an ELDATA-format data set named TEMP Type iset icase must be created via AUS/ELDATA before CONF is executed:

```
@XQT AUS
ELDATA: TEMP Type iset icase
G=1
E=1: D1 D2 - - END$  D(NO ITEMS) FOR ELEMENT 1, GROUP 1.
E=2: D1 D2 - - END$  D(NO ITEMS) FOR ELEMENT 2, GROUP 1.
-
-
-
```

The functions performed by SPAP processors in support of experimental elements are summarized below. None of the other SPAP processors (e.g. the plot programs) recognize experimental elements.

Processor	Function
ELD	Processes element definitions, and creates DEF, GD, GTIT, and DIP data sets.
TOPO	Accounts for presence of experimental elements in constructing the topological arrays, KMAP, AMAP, enabling system matrix assembly and factoring to be performed.
E	Creates the E-state data sets for experimental elements, in skeletal form. Adds contribution of experimental element weights to DEM.
EYS	Inserts the K, P, C, and D arrays into the E-state.
K	Includes experimental element K's in system K.
M	Includes experimental element CM's in system CEM.
KG	Includes experimental element KG's in system KG.
AUC	Permits loading of TEMP Type data sets via ELDATA.
EQNF	Adds contribution of experimental element fixed-joint forces to equivalent nodal force arrays.
GSF	Computes S for each element, including thermal effects, if present. S is embedded in the E-state, for use by KG, if requested via the EMBED reset control. S is not printed by GSF unless ONLINE=2. STPS Type data set are not produced by GSF for experimental elements. PSF does not recognize experimental elements.

The functions performed by SPAP processors in support of experimental elements are summarized below. None of the other SPAP processors (e.g. the plot programs) recognize experimental elements.

Processor	Function
ELD	Processes element definitions, and creates DEF, GD, GTIT, and DIP data sets.
TOPO	Accounts for presence of experimental elements in constructing the topological arrays, KMAP, AMAP, enabling system matrix assembly and factoring to be performed.
E	Creates the E-state data sets for experimental elements, in skeletal form. Adds contribution of experimental element weights to DEM.
EKS	Inserts the K, P, C, and D arrays into the E-state.
K	Includes experimental element K's in system K.
M	Includes experimental element CM's in system CEM.
KG	Includes experimental element KG's in system KG.
AUS	Permits loading of TEMP Type data sets via ELDATA.
EQNF	Adds contribution of experimental element fixed-joint forces to equivalent nodal force arrays.
GSF	Computes S for each element, including thermal effects, if present. S is embedded in the E-state, for use by KG, if requested via the EMBED reset control. S is not printed by GSF unless ONLINE=2. STPS Type data set are not produced by GSF for experimental elements. PSF does not recognize experimental elements.